Instituto Tecnológico de Costa Rica

Doctorado en Ciencias Naturales para el Desarrollo

Escuela de Ingeniería Electrónica

A methodology for automated design and implementation of complex analog and digital CMOS integrated circuits applying a genetic algorithm and a CAD tool for multiobjective optimization

Tesis para optar por el grado de Doctor

Roberto Pereira Arroyo

Cartago, May 19, 2014

Abstract

This dissertation proposes an automated methodology to design and optimize electronic integrated circuits, something that could be called simulation-driven optimization. The concept of Pareto optimality or the so called Pareto front is introduced as a useful analysis tool in order to explore the design space of such circuits. A genetic algorithm (GA) is employed to automatically detect this front in a process that efficiently finds optimal parameterizations and their corresponding values in an aggregate fitness space. Since the problem at hand is inherently a multi-objective optimization task, many different performance measures of the circuits must be able to be easily defined and computed as fitness functions.

The methodology has been validated through measurements of several fabricated test cases, using MOSIS fabrication services for a standard 0.5m CMOS technology.

Keywords: Optimization, CMOS, Genetic Algorithm, Pareto front, Integrated circuit

to Evelyn, Andrés, Sebastián and Elías

Acknowlegments

Many people have helped me during these years of work and deserve my deepest gratitude. Dr. Alfonso Chacón provided me with support, feedback and guidance. Dr. Pablo Alvarado, whom I shared seminal ideas and was always ready to give advice. Former students Leonardo Rivas, Frank Nicaragua, Dave Porras, Berny Dinarte, José Ibarra, Minor Coto and Roberto Molina spent many hours implementing test cases, trying different scenarios, which ultimately have provided part of the data presented in this dissertation. I also want to thank Dr. Renato Rimolo and Dr. Juan Luis Crespo for proofreading the draft document and giving valuable comments and suggestions.

Roberto Pereira Arroyo

Cartago, May 17, 2014

Contents

List of Figures

Table index

Chapter 1

Introduction

Analog integrated circuit design is a field that many consider as an art. In general, these kind of circuits contain an amount of parameters (variables) that the designer can adjust and, moreover, they have a direct impact on the perfomance of the circuit. Traditionally, analog design is made by hand using very general analytic equations in order to approximately describe the behavior of the circuit. Then, several (or many) simulations are performed to verify that the circuit works as expected. This is an iterative task and, usually, very time-consuming and furthermore, there is absolutely no guarantee about how optimum the resulting circuit is, for example, in terms of Silicon area, power consumption, *slew rate*, gain, offset, etc.

Another aspect that must be taken into consideration is the fact that usually it is needed to meet more that one design metric. This turns the task of exploring a given circuit's design space into a multiobjective optimization problem. Thus, the basic problem is to find a trade off among design metrics: one cannot improve one without having a negative effect in the others. Fig. 1.1 shows a design heptagon which illustrates, with typical metrics, this trade off [48]

This dissertation introduces an automated optimization strategy that can be used for the fast and efficient design of CMOS electronic structures such as Operational Transconductance Amplifiers (OTAs), current supplies, MOS Current Mode Logic (MCML) gates and other analog and digital building blocks, taking advantage of the power and versatility of Genetic Algorithms (GAs). This not only significantly reduces circuit's design time, but also ensures efficient and robust solutions. Some approaches [36], [29] tie the optimization problem to the specific topology of a circuit and to its parameters, making necessary a relative exhaustive search of the parameter space. Genetic Algorithms, on the other hand, work at a higher level of abstraction in which specific information about the circuit being optimized is not required: the GA only receives a set of fitness values (e. g. real numbers), representing circuit metrics such as power consumption, area, slew rate, etc. The proposed optimizer uses the genetic algorithm named PESA (Pareto Envelope-based Selection Algorithm) [11], and relies on a standard circuit simulator (e. g. Eldo™, Spectre™, SPICE and alike) to deal with the complexity of physical MOS transistor parameters and circuit topology. Circuit parameters

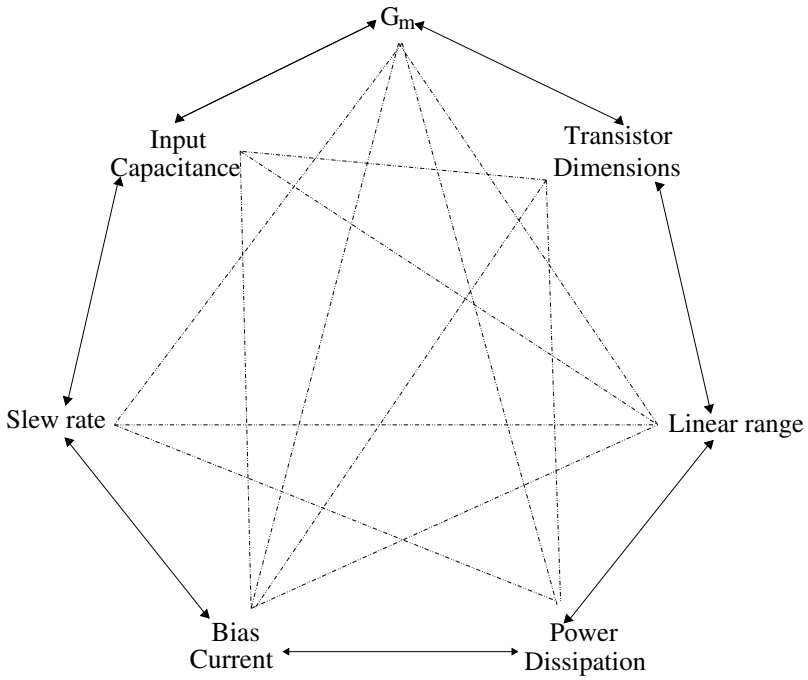


Figure 1.1: An example of the trade-off between design metrics.

like supply voltages, bias currents, width and length of transistors, etc., are generated by the genetic algorithm and given to the simulator, where the computation of the fitness values takes place. Thus, the designer can change either the optimization algorithm or the simulation models without much effort. Analog Design Automation has been a research topic for over 20 years [28]. Since then, there has been a continuous effort in order to develop this area [14, 26, 23]. Following this trend, we contribute by taking specific application examples and showing the usefulness of GAs, multi-objective optimization and the Pareto front for automating analog and digital design.

The use of genetic algorithms is not new, in fact there is a wide range of engineering applications where they have been succesfully used. Optimization of airplane wing planform design [44], manufacturing systems and mobile robots are few examples [19]. Although there are relatively recent applications of these algorithms to submicron CMOS circuit design and electronic design in general [33],[51], these usually not consider the problem as multiobjective. In addition, most of research efforts focus in devolping and studying the optimization algorithms themselves [14, 40, 50], leaving aside the subject of circuits design as application examples, sometimes using relatively simple circuits. There seems to be more interest in assessing algorithm performance that their utility for circuit design. The latter is our main focus on the other hand, particularly exploring circuit desing space in the extreme of very low power consumptions (in the nanowatt order) and using technologies with high variability.

The proposed methodology is applied to solve problems like very-low-power gunshot detection circuitry [9], a self-biased low-power current source and several submicron MCML gates [47].

1.1 Goals and methodology

The general objective of this dissertation is to develop an automated methodology for designing and implementing complex analog and digital CMOS circuits, using multi-objective optimization CAD tools.

The following are specific goals of this dissertation:

1. To validate a genetic optimization tool that can be integrated into a commercial submicron CMOS IC design flow.
2. To propose a circuit performance evaluation framework in order to feed the automated design space exploration and subsequent Pareto front generation.
3. To propose a framework for the generation and evaluation of the design space of submicron CMOS ICs, including the effect of process variability on the optimization.
4. To prove the efficiency of the methodolgy by implementing and verifying at least three complex CMOS analog and digital cases.

In order to achieve the aforementioned specific goals, a number of tasks or problems are required to be solved, namely:

1. To establish a solid software interface between the algorithmic implementation and a commercial SPICE simulator. This interface has to be independent of the Genetic Algorithm and also should not be bound to any specific simulator.
2. The required circuit performance computation routines for integrating the tool into a commercial simulator must be implemented.This means that, starting from the standard functions of a simulator, more complex functions must be derived in an automated way.
3. A Pareto front can have more than three dimensions. Thus a straightforward analysis routine must also be implemented in order to quickly process the results, when the front is four-dimensional or higher.
4. A variability model, previously justified and verified, must be integrated into the tool, such that not only more robust designs can be obtained but also ones whose precision can be predicted.
5. The effectiveness of the methodology will include three complex test cases, namely:
   - An low-power analog signal processing circuit for gunshot detection in forest environments.
   - An low-power, temperature and voltage variation-tolerant current source.
   - A family of high-speed MCML cells.

   The verification includes fabrication, circuit characterization through measurements and contrast with results from non-optimized prototypes or other examples reported in the literature.

Considering the fact than optimization is included in a rather simplistic way in commercial simulators, the methodology presented in this dissertation should serve as a bridge between two apparently far-separated disciplines: integrated circuit design and algorithmic optimization, showing how the interplay between them can be of great help to circuit designers.

1.2 Thesis outline

Chapter 2 introduces heuristic optimization methods, such as Genetic Algorithms, how they work as well as the architecture of the electronic design automation (EDA)tool developed as part of this dissertation.

Chapter3 elaborates on examples of typical problems faced in classic, hand-made CMOS analog design, along with the significant improvement in the design process and the final results obtained after using the proposed Genetic CAD tool.

Chapter 4 shows the process followed in the development of a test platform for the fabricated application specific integrated circuits (ASIC) as well as the corresponding measurement results.

Chapter 5 defines the strategy to be applied for the analysis of variability in selected designed structures, based on the model proposed by Pelgrom [46].

Chapter 6 presents the conclusion of this work.

Chapter 2

Heuristic methods for multi-objective optimization

Heuristic optimization methods, such as Genetic Algorithms, are essentially computational and therefore have been increasingly applied following the development of electronic computing devices [25]. The heuristic optimization paradigm is based on concepts found in nature; for example, the principle of evolution through selection and mutation (genetic algorithms), the annealing process of melted metals (simulated annealing) or the self-organization of ant colonies (ant colony optimization). What is expected in general from a heuristic? First, a heuristic should be able to provide high quality (stochastic) approximations to the global optimum, but it is not supposed to give the exact solution to the problem. Second, a well behaved heuristic should be robust to changes in problem characteristics, i. e. it should not only fit a single problem instance, but a whole class of problems. Third, a heuristic should be easily implemented regardless of any arbitrary number of problem instances and, even if the heuristic is stochastic, it should not contain subjective elements. Given the above definition of heuristics, one of their major advantages consists in the fact that their application does not rely on a set of strong assumptions about the optimization problem; it suffices the possibility to evaluate the objective function for a given element of the search space. No assumption on some global property of the objective function is required, nor the computation of its derivatives (common in classic optimization strategies). Therefore, the problem of optimizing analog and digital integrated circuits meets many if not all of the requirements that make a problem a good candidate for a heuristic solution.

2.1 Current approaches for sizing and optimization of integrated circuits

As briefly stated before, various methods have been published for calculating the performance features of CMOS digital and analog integrated circuits. These methods offer a broad range of possibilities depending on the way the circuit performances are obtained, wether or not

transistor mismatch is considered and, mainly, what kind of algoritm is used to solve the optimization problem (deterministic or heuristic, global or local optimizer).

A deterministic optimization approach, shown in [30] is called Geometric Programming (GP).

A geometric program is an optimization problem of the form:

$$\begin{aligned} &\text{minimize} && f_0(x), \\ &\text{subject to:} && f_i(x) \leq 1, \quad i = 1, \ldots, m, \\ &&& g_i(x) = 1, \quad i = 1, \ldots, p, \\ &&& x_i > 0, \quad i = 1, \ldots, n. \end{aligned} \tag{2.1}$$

where $f_0, \ldots, f_m$ are posynomial functions and $g_0, \ldots, g_p$ are monomial functions. Although the optimization process is very fast, the GP can only optimize over a convex function and the performances and constraints must be expressed as posynomial functions. This is a task that can be time-consuming and it is bound to a specific circuit topology. In the end, the accuracy of this simplified circuit's model must be verified against Spice simulations. While geometric programming is certainly known, it is nowhere near as widely known as, say, linear programming. In addition, advances in general-purpose nonlinear constrained optimization algorithms and codes have contributed to decreased use (and knowledge) of geometric programming [30].

Another approach, Linear programming (LP) is an optimization method that has been used since World War II. In LP the objective function is linear and all constraints can be supplied as linear equalities or inequalities. More formally, a LP can be expressed as:

$$\begin{aligned} &\text{maximize} && \mathbf{C}^T\mathbf{x}, \\ &\text{subject to:} && \mathbf{A}\mathbf{x} < \mathbf{b} \end{aligned} \tag{2.2}$$

Here, $\mathbf{x}$ represents the decision variables of the LP, which are the design parameters. The $\mathbf{C}^T\mathbf{x}$ objective function is some linear function of $\mathbf{x}$ that is determined by what we are trying to optimize (e.g.: gain,bandwidth, slew rate, etc.). The equations $\mathbf{A}\mathbf{x} < \mathbf{b}$ represent the set of constraints which essentially relate some linear transformation of $\mathbf{x}$ to a constant $\mathbf{b}$. It should be noted that any inequality or equality constraint can be put into this form. There are several algorithms that are able to solve LPs and its variants in an efficient manner, the Simplex or Dantzig algorithm being one of the most popular [13].

In similar way as in Geometric Programming, the Linear Programming method requires an understanding of circuit topologies and equations for the objective functions of interest, i.e. a linear model of each specific circuit must be derived. For example in [8] the algorithm requires an objective function, upper and lower bounds for all voltage nodes, and a set of inequalities that force all transistors to operate in saturation.

Simulated Annealing is a stochastic method that has been used for analog circuit optimization for over 20 years [24, 37].

Annealing is the process of heating a solid until thermal stresses are released. Then, in cool-

ing it very slowly to the ambient temperature until perfect crystals emerge. The quality of the results strongly depends on the cooling temperature. The final state can be interpreted as an energy state (crystaline potential energy) which is lowest if a perfectly crystal emerged. If we compare optimization to the annealing process, the attainment of a global optimum is analogous to the attainment of a good crystal structure. Simulated annealing was introduced in [34]. Fig. 2.1 shows the correspondig pseudocode. The core of the algorithm is the *Metropolis* procedure, which simulate the annealing process at a given temperature T [38].*Metropolis* also receives the current solution S which it improves thorough local search. It must also be provided with the value M, which is the amount of time that annealing must be applied, at a given temperature. The procedure Simulated_annealing simply invokes *Metropolis* at various (decreasing) temperatures. Temperature is initialized to a value T_0 at the beginning of the procedure, and is slowly reduced in a geometric progression; the parameter α is used to achieve this cooling behavior. The amount of time spent in annealing at a temperature is gradually increased as temperature is decreased. This is done using the parameter $\beta > 1$. The variable *Time* keeps track of the time being expended in each call to the *Metropolis*. The annealing procedure halts when *Time* exceeds the allowed time.

Algorithm: Simulated Annealing

1: initialize $T = T_0$ $S = S_0$ $Time = 0$
2: **repeat**
3: Call *Metropolis*(S,T,M)
4: $Time = Time + M$
5: $T = \alpha \times T$
6: $M = \beta \times M$
7: **until** $Time \geq MaxTime$
8: Output best result

Figure 2.1: Pseudo code for a Simulated Annealing algorithm.

Fig. 2.2 shows the *Metropolis* procedure. It uses the procedure *neighbor* to generate a local neighbor *NewS* of any given solution S. The function*Cost* returns the cost of any given solution S. If the cost of the new solution *NewS* is better than the cost of the current solution, then certainly *NewS* is an acceptable solution, and therefore it is set $S = NewS$. If the cost of the new solution is worse than the current solution,*Metropolis* will accept the new solution on a probabilistic basis. A random number is generated in the range 0 to 1. If this random number is smaller than $e^{-\Delta h/T}$, where Δh is the difference in costs and T is the temperature, then the inferior solution is accepted. This criterion for accepting the new solution is known as *Metropolis criterion*. The *Metropolis* procedure generates and evaluates M solutions. Notice that at high temperatures the probability of accepting inferior solutions is high. This characteristic is what avoids simulating annealing from being trapped in local minima. On the contrary, as temperature decreases the probability $e^{-\Delta h/T}$ falls to 0 [49]. This situation represents the end of the annealing process and the algorithm proceeds in a

greedy-fashion towards the minimum.

Algorithm: *Metropolis*(S,T,M)

1: **repeat**
2: $NewS = neighbor(S)$
3: $\Delta h = Cost(New(S)\text{-}Cost(S)$
4: **if** $\Delta h < 0$ or $random < e^{-\Delta h/T}$ **then**
5: $S = NewS$
6: **end if**
7: $M = M - 1$
8: **until** $M = 0$

Figure 2.2: Pseudocode for the *Metropolis* procedure

Although, as already stated, simulating annealing has been around for many years, these approaches generally treat the multiobjective problem by deriving a circuit-specific equation which hopefully will reflect the effect of single objectives. In other words, the concept of Pareto optimality is not used. Related approaches like the one reported in [22] relies on the fact that optimization carried out over a metamodel (which is an abstracted representation of the circuit model) instead of the actual circuit will allow fast design space exploration and reduce the design cycle time. In that paper three different optimization algorithms are compared: exhaustive search, tabu search and simulated annealing algorithms are analyzed to determine their suitability for metamodeling-based optimization, however their results are not necessarily applicable for transistor-level simulation as it is the approach presented in this dissertation.

2.2 Genetic Algorithms

Genetic algorithms imitate the evolutionary process of species that sexually reproduce. New candidates for the solution are generated with a mechanism called crossover which combines part of the genetic material of each parent and then applies a random mutation. If the new individual, called child, inherits good characteristics from its parents it will have a higher probability to survive. The fitness of the child and parent population is evaluated in function **survive** (statement 10 in pseudo code shown below) and the survivors can be formed either by the last generated individuals $\mathcal{P}''$, $\mathcal{P}'' \cup \{\text{the fittest from } \mathcal{P}'\}$, only the fittest from $\mathcal{P}''$ or the fittest from $\mathcal{P}' \cup \mathcal{P}''$. A pseudo code for genetic algorithms is shown in Fig. 2.3.

The genetic algorithm first accepts a set of solutions (statement 3) and then constructs a set of neighbor solutions (statements 4–10). In general, a predefined number of generations provides the stopping criterion [25].

Algorithm: Genetic algorithms.

```
1: Generate initial population 𝒫 of solutions
2: while stop criterion not met do
3:    Select 𝒫′ ⊂ 𝒫 (mating pool), initialize 𝒫″ (set of children)
4:    for i = 1 ...n do
5:       Select individuals x_a and x_b at random from 𝒫′
6:       Apply crossover to x_a and x_b to produce x_child
7:       Randomly mutate produced child x_child
8:       𝒫″ ← 𝒫″ ∪ x_child
9:    end for
10:   𝒫 ←survive (𝒫′, 𝒫″)
11: end while
```

Figure 2.3: Pseudo code for a genetic algorithm.

2.2.1 Circuit Performance Evaluation using the Pareto Front

The aggregate fitness function F for a circuit A with the parameterization $\mathbf{u}$ is defined as:

$$F(A_{\mathbf{u}}) = \Phi(f_1(A_{\mathbf{u}}), f_2(A_{\mathbf{u}}), \ldots, f_n(A_{\mathbf{u}})) \tag{2.3}$$

With the individual fitness functions $f_i(A_{\mathbf{u}})$ defined to increase monotonically with the fitness of some particular aspect of the circuit's behavior. For example, if the objective is to optimize an OTA (Operational Transconductance Amplifier), fitness values like linear range and the slew rate may be directly related. On the other hand, the input capacitance and the transconductance are inversely related to preserve the condition of increasing monotonicity. The functions f_i span a multidimensional fitness space, where each point represents the performance of a circuit parameterized with one point $\mathbf{u}$ in a parameter space. The general form of Φ is assumed unknown, but it has to increase monotonically with increasing values of all fitness functions f_i. This condition ensures that a point in the fitness space can be considered fitter than all other points with smaller values in all dimensions. In Fig. 2.4, for example, the point q_1 is fitter than the point q_4 and all other elements within the gray rectangle. In this context, the point q_1 is said to dominate q_4. All non-dominated points in a set define the Pareto front of that set. In the example of Fig. 2.4 this front is defined by the points q_1, q_2, q_3. Choosing a parameterization that is not in the front is always a bad choice, since there is another point on the front with a better aggregate fitness. The previous concepts can be expressed mathematically using the following equation:

$$\hat{P} = \{\langle \mathbf{u} \in \mathbb{P}_A, \mathbf{f}(A_{\mathbf{u}})\rangle | \neg\exists \mathbf{v} \in \mathbb{P}_A : \mathbf{f}(A_{\mathbf{v}}) \succ \mathbf{f}(A_{\mathbf{u}})\} \tag{2.4}$$

where $\hat{P}$ is the Pareto front, $\mathbf{f}$ is the vector of fitness functions $[f_1, ..., f_n]^T$ and $\mathbb{P}_A$ is the parameter space of circuit A. The partial order relationship '$\succ$' on $\mathbf{f}$ describes the domination property and is defined as:

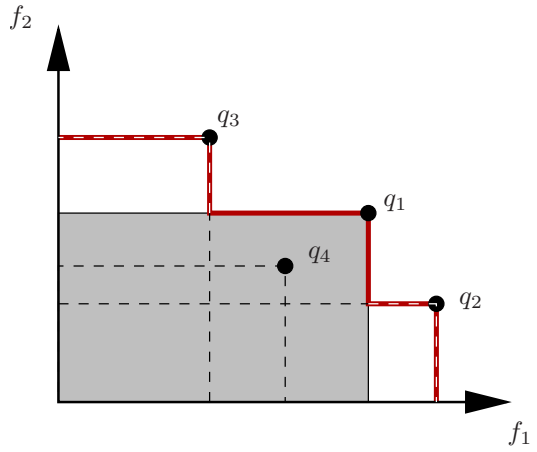


Figure 2.4: Pareto front in a two-dimensional fitness space.

$$\begin{aligned} f(A_{\mathbf{v}}) \succ f(A_{\mathbf{u}}) \Leftrightarrow \\ \forall i : f_i(A_{\mathbf{v}}) \geq f_i(A_{\mathbf{u}}) \wedge \exists i : f_i(A_{\mathbf{v}}) > f_i(A_{\mathbf{u}}) \end{aligned} \tag{2.5}$$

The evaluation process can therefore be considered as a mapping process that transforms the valid parameter space $\mathbb{P}_A$ into a connected region in the fitness space $[f_1, \ldots, f_n]^T$. The Pareto Front is the border of this region delimited by the partial optima [40]. Any algorithm that finds the Pareto front for a set of fitness points implements eqs. (2.4) and (2.5). Since the parameter space $\mathbb{P}_A$ usually contains an infinite number of parameterizations, the next problem consists in choosing a representative set of samples from $\mathbb{P}_A$, such that their Pareto front can be assumed to be a reliable approximation of the exact front extracted for the complete space. A naive approach would be to regularly sample the values of each parameter, since the number of necessary evaluations will increase exponentially with the number of parameters. For example, a circuit with seven parameters (design variables), each sampled five times, would require $5^7 = 78125$ evaluations. To avoid this brute-force parameter search, here the multi-objective evolutionary algorithm PESA is employed. This genetic approach suppresses the computation of useless parameterizations and concentrate the analysis on those regions of the parameter space that provide promising results. Even if this algorithm also discretizes the parameter space, through a numeric representation with a finite number of bits, the resolution used for each parameter can be as high as necessary, without the menace of an exponential explosion of the search space. The number of evaluations required is then proportional to the number of bits used to represent a parameterization.

2.3 Pareto Envelope-based Selection Algorithm

The Pareto front has been defined as the subset of parameterizations (or *individuals*) that are *non-dominated* in a fitness space. Several algorithms have been proposed for its computation, for instance, SPEA (Strength Pareto Evolutionary Algorithm [52]) and PAES (Pareto Archived Evolution Strategy [35]). The PESA algorithm (**P**areto **E**nvelope-based **S**election

Algorithm) by Corne et al. has been chosen in this work to seek for the Pareto front, since it has proven to perform better than other available techniques [11].

PESA is an evolutionary algorithm. The search for the non-dominated front relies on the principles of mutation and crossover of the currently fittest individuals. Mutation tries to improve a parameterization through a few random changes. It searches for fitter candidates in the neighborhood of previously found solutions. Crossover takes two parent candidates and combines them in order to generate a third, possibly fitter individual, where the combination makes it possible to sample a larger region of the parameter space.

In the PESA implementation, a parameterization (also *phenotype*) needs a binary representation (*chromosome*). The mutation process inverts the value of a chromosome's bit if a random number drawn from a uniform distribution between 0 and 1, is smaller than the desired mutation rate P_m. A uniform crossover technique is also used, in which each bit of the child is inherited with the same probability from each parent.

The algorithm administrates two sets of phenotypes, called *populations*. The *external* population $\mathcal{P}_E$ represents the current approximation of the Pareto front. The *internal* population $\mathcal{P}_I$, usually smaller, contains a set of new candidates to be eventually included in the front. To avoid the external front exceeding a predefined maximal size, some old elements may have to be removed. The selection of these individuals is the main difference between most Pareto evolutionary algorithms. PESA keeps track of the degree of crowding at different regions of the fitness space. It selects for removal those elements in the most dense sections, such that the phenotypes in the front tend to be equally distributed. The algorithm is outlined in Fig. 2.5.

The incorporation of non-dominated candidates into the Pareto front at line 4 includes the removal of all individuals that are dominated by the new incomers. This is necessary to maintain the consistency of $\mathcal{P}_E$. For the required crowding measure, Corne et al. have originally suggested to partition the fitness space in regular hyper-boxes. A "squeeze factor" is then assigned to each box, defined as the total number of phenotypes within the box. This histogram-based density estimation is employed at line 6: an individual in the box with the highest squeeze factor is selected for removal. The opposite occurs in the choice of individuals for crossover and mutation (lines 11 and 13): a binary tournament strategy is used to direct the attention towards the least dense regions of the front, i. e. from a randomly chosen pair of individuals, the one with the smallest squeeze factor is always taken as parent, breaking ties randomly. Both actions help keeping the parameterizations equally distributed in the fitness subspace spanned by the Pareto front.

The crossover probability P_c at line 10 defines the fraction of new individuals that are generated through crossover. The probability $1 - P_c$ specifies, therefore, the number of parameterizations obtained from mutation of a single parent.

Everingham et al. [20] replace the histogram in the squeeze factor computation with a kernel density estimator that uses Gaussian kernels with a diagonal covariance matrix. This method has also been adopted here. The variances of the kernel at each dimension are fixed to a fraction of the bounding box that delimits the known fitness space.

Algorithm: PESA [11]

initialize the external population $\mathcal{P}_E$ with the empty set
initialize the internal population $\mathcal{P}_I$ with n_I random individuals
repeat
  find all individuals in $\mathcal{P}_I$ that are not dominated by any element of $\mathcal{P}_I \cup \mathcal{P}_E$ and incorporate them into $\mathcal{P}_E$
  while $|\mathcal{P}_E| > n_E$ **do**
    select and remove an individual of $\mathcal{P}_E$
  end while
  remove all remaining elements of $\mathcal{P}_I$
  while $|\mathcal{P}_I| < n_I$ **do**
    if probability P_c **then**
      select two parents from $\mathcal{P}_E$ and produce single child by crossover and mutation
    else
      select single parent from $\mathcal{P}_E$ and produce single child by mutation
    end if
    add child to $\mathcal{P}_I$
  end while
until maximum number of iterations is reached
return $\mathcal{P}_E$

Figure 2.5: Pareto Envelope-based Selection Algorithm

Here, an additional extension originally proposed in [1] has been "borrowed" from the simulated annealing optimization techniques. The mutation rate is allowed to decrease asymptotically from an initial value $P_{m_{initial}}$ towards the desired final rate $P_{m_{final}}$, resembling the temperature reduction typically found in such algorithms:

$$P_m = (P_{m_{initial}} - P_{m_{final}}) \exp(i/\tau) + P_{m_{final}} \tag{2.6}$$

with i the iteration number and τ the mutation decrease factor. The implementation has been integrated in an open source software library described in [16]. The increased mutation rates at the beginning stimulate a stronger random sampling of the parameter space. At early iteration stages the points in the front have not suffered a long evolution, and thus their parameter values are still relatively unstable. The random sampling accelerates the localization of fitter candidates. As soon as several iterations have confirmed the fitness of the points in the front, the random sampling becomes rather harmful to the process. Lower mutation rates give more weight to the information contained in the parents, which are at later iterations probably fitter than random candidates.

The most important parameters for the PESA algorithm and their corresponding default values (used in the evaluation sections) are listed in Table 2.1.

Table 2.1: PESA Parameters and Typical Values

Symbol	Parameter	Value
n_I	Size of the internal population $\lvert\mathcal{P}_I\rvert$	10
n_E	Size of the external population $\lvert\mathcal{P}_E\rvert$	100-200
P_c	Crossover probability	0.7
$P_{m_{initial}}$	Mutation rate	3/chromosome size
$P_{m_{final}}$	Mutation rate	1/chromosome size
τ	Mutation decrease factor	40
	Maximum number of iterations	500-1500
	Kernel size as fraction of the bounding-box size	1/32

Figs. 2.6 and 2.7 show the chromosome representation for a NAND gate and the OTA respectively. Bit strings were chosen to represent these chromosomes, defining minimum and maximum values for every parameter within the bit string. For example, in Fig. 2.7, the parameter I_b (Bias current) uses 5 bits which enables 32 possible distinct values for this parameter.

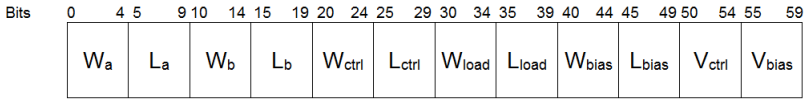


Figure 2.6: NAND gate chromosome

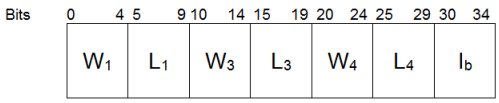


Figure 2.7: OTA chromosome

2.3.1 CAD tool architecture and its implementation

The aim of our tool is to generate the Pareto front of a given cell, i. e. the set of all non-dominated parameterizations, since they represent the best individuals we are looking for. This information can be used later on to find the desired operating point of a circuit (choosing a specific individual), depending on the amount of resources available for the designer. A high-level block diagram of the tool is shown in Fig. 2.8. The functionality of the tool is divided into two well-defined and independent processes: a circuit representation phase and an optimization phase.In the current version of our tool, the circuit representation is captured with a SPICE-like netlist and fed to a circuit simulator such as Eldo^{TM} or Spectre^{TM}, where the computation of the specified performance metrics takes place based on a well adjusted model of the process in which the circuit is to be fabricated (typically, and advanced version

of BSIM[1] with models provided from the semiconductor foundry). On the other hand, the core of the optimization process is based on the LTI-Lib [16], an open source library originally intended for image processing research [16], which provides the implementation of the PESA algorithm and the generation of the Pareto front. In Fig. 2.8, the thick dotted line illustrates this separation of tasks. Furthermore, the optimization and representation processes are communicated through TCP/IP sockets, which allows for each process to reside on different machines or operating systems.

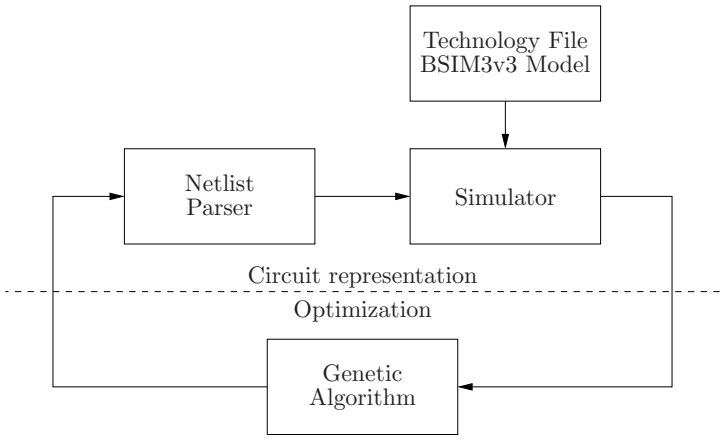


Figure 2.8: Optimization CAD tool block diagram.

Fig. 2.9 presents a more detailed illustration of the steps followed in order to implement the optimization methodology.

Figs. 2.10 and 2.11 present flow diagrams for some of the routines that were implemented in order to compute different fitness functions. All of these routines process the output file generated after each simulation and basically search for a specific feature, for instance a node capacitance, or use the given data to calculate values like linear range or the transconductance.

[1]Berkeley Short-channel IGFET Model

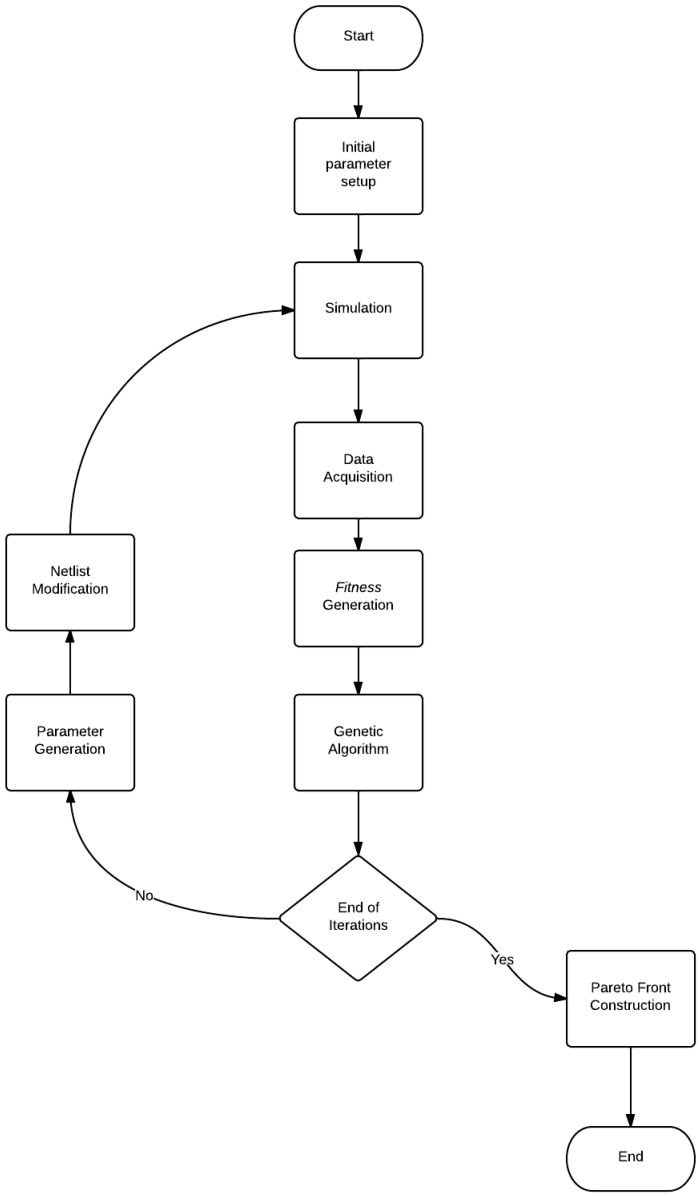


Figure 2.9: General optimization methodology implementation flow diagram

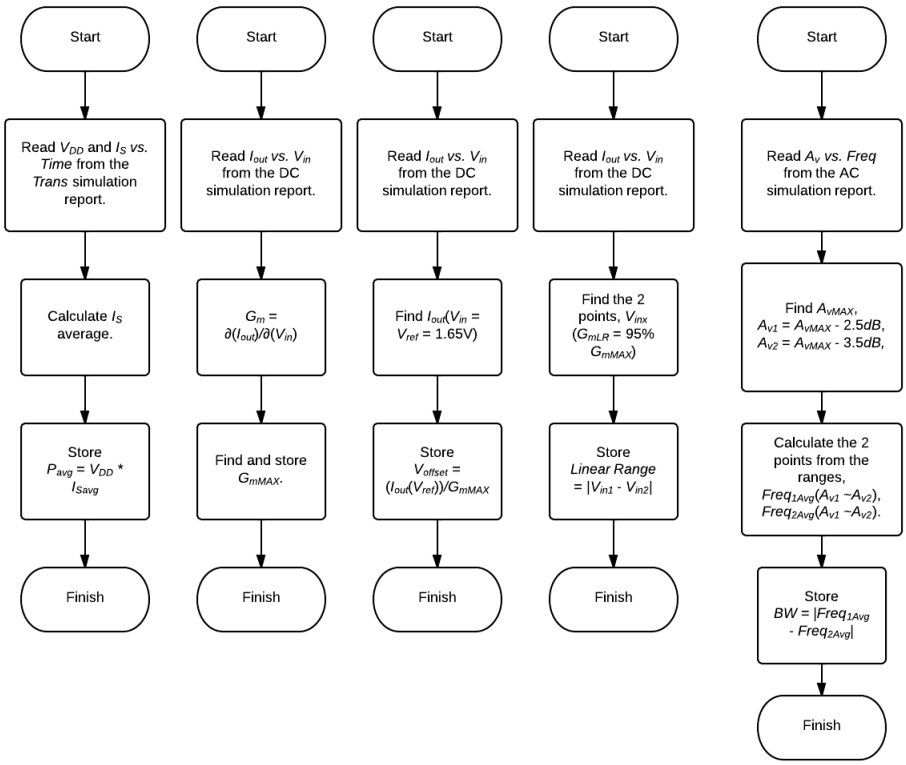


Figure 2.10: Flow diagrams of fitness functions implementations I

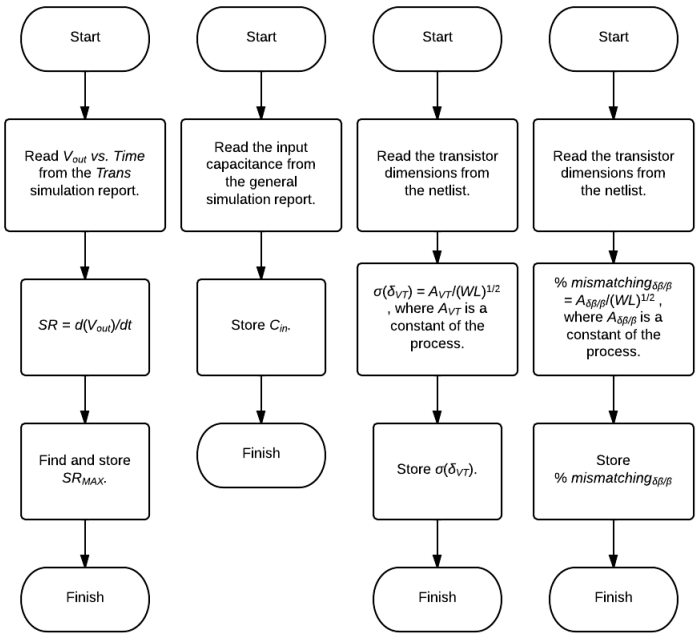


Figure 2.11: Flow diagrams of fitness functions implementations II

Chapter 3

Application of automated methodology to CMOS circuit design

An optimized design of particular electronic cells usually kicks-off with a first-hand design stage based on a very approximate first order model, something that requires multiple iterations through a tiring process of hand re-calculation, simulation and fitting, in order to reach a limited set of specifications. The use of a heuristic tool, not only cuts down the process, since the designer is not forced to re-calculate using the first order model again, but it also allows for an increase in the set of specifications that usually require mutually exclusive goals that first order models are typically ill suited to reconcile. This case is particularly well illustrated in the case of submicrometric CMOS design, where the transistor models can easily become cumbersome, especially when trying the cover all the MOSFET regions of operation, from weak to strong inversion, either in saturation or in linear mode. Thus, for long channel MOSFET design, the use of a first order model such as the EKV [18] or ACM [21] models is limited to the the initial specification of the cell under design, a specification that is not necessarily enforced in this first stage, as the tool will be used to solve this task. And of course, in the case of short-channel design –where accurate first-hand models are largely unwieldy–, the designer can very well kick-start the process with an ill-fitted approximation, knowing that at worst, the optimization process will only require some extra iteration time.

In this chapter, examples of typical problems faced in classic, hand-made CMOS analog and digital design are shown, along with the significant improvement in the design process and the final results obtained after using the proposed Genetic CAD tool, results that in many cases not only comply, but surpass the critical specifications given to the problem. In most of the cases, results are compared with published data of similar hand-made designs.

3.1 Optimization of MOS Current Mode Logic: a proof of concept

MCML is a circuit technique that has been used in applications of high-speed, mixed signal environments due to its reduced switching noise, immunity to common-mode noise and, especially, because its power consumption does not increase with the frequency of operation[39], whereas in standard CMOS circuits, power consumption increases linearly with frequency. In[42] it is shown that MCML dissipates less power at operation frequencies of more than 300 MHz. More recently, it has also been shown that subthreshold MCML can be used to implement digital circuits at frequencies below a few megaherz, with a better power-delay product than CMOS counterparts[7]. However, designers have been reluctant to use MCML instead of CMOS due to the complexity of MCML and the lack of automation tools. This situation has made impossible to produce robust and power efficient designs at low cost and reasonable time to market[41]. The fundamental MCML inverter/buffer is shown in Fig. 3.1. MCML circuits have three main components: PMOS transistor loads, one or more differential pairs depending on the number of logic inputs and a constant current source, controlled by the voltage V_{bias}. All logic inputs and outputs are fully differential. The circuit operation is based on current steering, i. e. the tail current produced by the transistor M_{bias} is steered into one of the branches depending on the differential inputs. This current develops a resistive voltage drop at the active load of the conducting branch, while in the non conducting branch the output voltage is pulled to V_{dd}, thus producing complementary outputs. For a single logic gate, its delay and power are respectively given by [42]:

$$D_{MCML} = C(\Delta V/I), \tag{3.1}$$

$$P_{MCML} = I \cdot V_{dd}, \tag{3.2}$$

where C is the load capacitance, I is the tail current and ΔV is the output voltage swing. Equation (3.1) indicates that the propagation delay can be reduced by lowering the voltage swing, decreasing the load capacitance or increasing the tail current. However, from Eq. (3.2) it is seen that increasing the tail current directly impacts the power consumption.

If the circuit shown in Fig. 3.1 is operating in the mid swing point of its voltage transfer curve, the currents in the two branches are equal to $I/2$, both transistors in the differential pair are in saturation and their currents can be expressed as[5]:

$$\frac{I}{2} = \frac{\mu_0 C_{\text{ox}}(W/L)_A}{2} \cdot \frac{(V_{GSA} - V_t)^2}{1 + (U_d + \frac{V_{GSA}-V_t}{E_c L})} \tag{3.3}$$

where U_d is the mobility degradation coefficient, E_c the critical electric field for velocity saturation, μ_0 the permeability of vacuum, C_{ox} the oxide capacitance per unit gate area, V_t the threshold voltage and $(W/L)_A$, V_{GSA} are the width, length and gate-source voltage for transistors M_A, respectively.

MCML design is a complex task as its objective metrics are interdependent, and numerous circuit parameters have an effect on these metrics. In the example shown in Fig. 3.1, nine

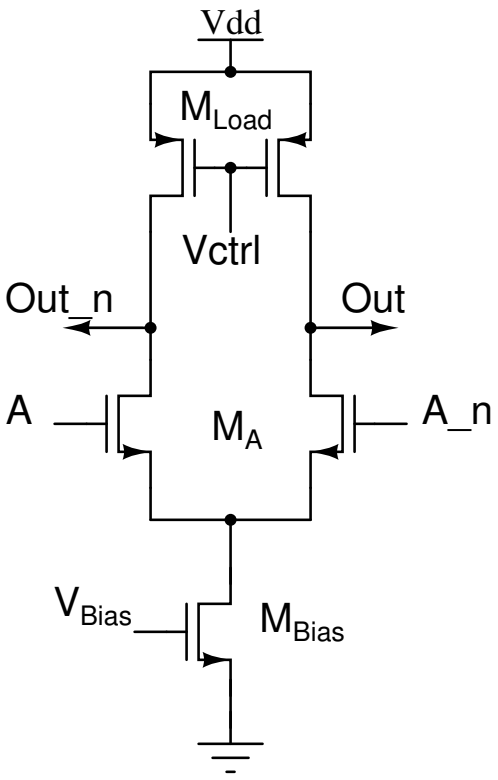


Figure 3.1: MCML inverter/buffer. Transistors at the differential pair and at the active loads have identical dimensions. Therefore their naming as "M_A" and "M_{Load}", respectively.

circuit parameters can be varied by the designer, namely: (W, L) for M_{bias}, M_a and M_{load} plus V_{dd}, V_{ctrl} and V_{bias}.

Fig. 3.2 shows an example of a 2-level MCML circuit. It has been shown in[29] how optimizing 2-level gates becomes even more complex, as more parameters come into play, making its design by hand calculations a task of little practical use. Therefore it is apparent the need for an optimization strategy that is automated and independent of circuit topology.

Fig. 3.3 shows the Pareto front of a MCML Xor. Three design metrics were defined for this example: voltage swing, average power consumption and propagation delay, although as previously stated, other metrics can also be defined.

Table 3.1 shows a list of some selected results from the Pareto front. Each column in the table contains the set of parameters defined for the corresponding logic gate.

It is necessary to point out that the optimization methodology is simulation-driven at the schematic level. The layout is not included within the optimization, mainly because there are not tools for fast automatic layout generation of analog cells, at least they were not available at the time of this research. The fact is that analog design still remains a mainly hand-made task. However, post-layout simulation results are presented as thay are supposed to be close to a fabricated circuit.

Table 3.2 contains figures (fitness values in GA terminology) from post-layout simulations.

Fig. 3.4 shows the Pareto front of a Nand gate in CMOS 90 nm technology.

Table 3.1: Extraction of parametrizations for several MCML gates

Parameter	Xor	CarryOut	Nand	DFlip-Flop
Vdd (V)	2.7	2.7	2.7	2.7
Vbias (V)	2.7	2.7	2.7	2.7
Vcntrl (V)	0	0	0	0
Wbias(μm)	2.9	4.8	4.1	5.83
Lbias(μm)	0.65	1.5	1.22	0.43
Wa(μm)	4.8	2.58	3.45	3.95
La(μm)	2.48	0.75	1.22	0.43
Wb(μm)	2.9	5.15	3.1	5.65
Lb(μm)	0.75	0.65	0.9	1.53
Wc(μm)	5.14	4.2	-	-
Lc(μm)	1.30	0.51	-	-
WLoad(μm)	6	4.8	5.83	5.83
Load(μm)	2.72	0.83	0.75	0.83

Table 3.2: Performance measures of MCML gates, as obtained from post-layot simulations

Gate	Power (mW)	OutSwing (V)	Delay (ns)
Xor	0.4	1.45	1.3
CarryOut	0.7	0.5	0.96
Nand	0.78	0.7	1.7
DFlip-Flop	0.97	1.46	0.25

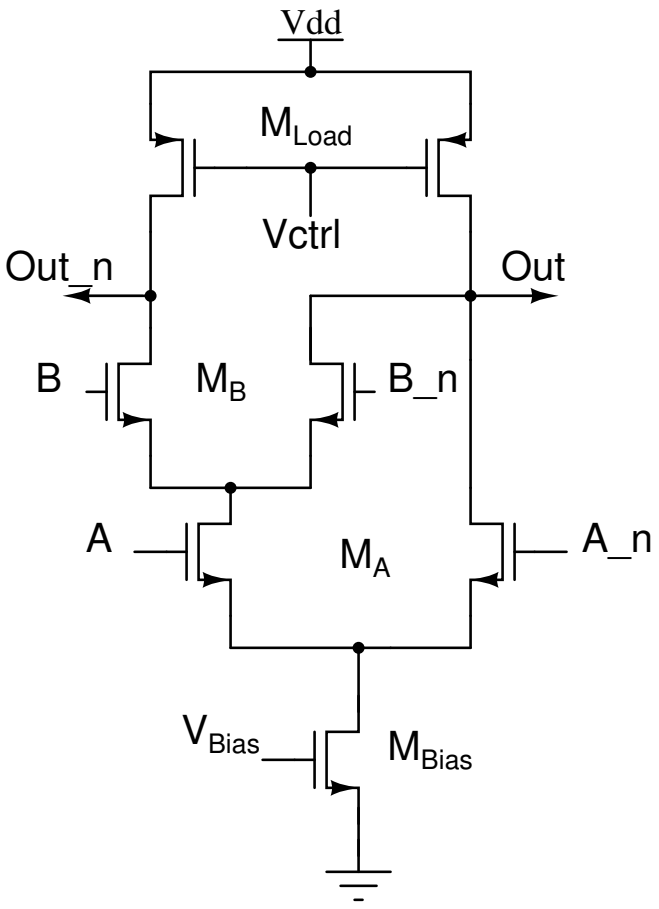


Figure 3.2: An example of a 2-level MCML gate, implementing both Nand/And logic functions.

Fig. 3.5 shows the timing simulation.

Fig. 3.6 shows the simulation in order to determine gate delay.

Fig. 3.7 shows the average power.

Fig. 3.8 shows the voltage swing.

3.2 A system for gunshot and chainsaw detection

The development of components of a wireless sensor network (WSN) for protection of tropical forests has been proposed in[2]. The ultimate goal is to detect gunshots (illegal hunting) and chainsaw noises (illegal timbering). The network's sensor nodes must be deployed in remote locations, be almost maintenance free and powered from small batteries. Therefore, low power consumption is critical as the battery charge must last for long periods of time. A very low power ASIC[1] implementation has been chosen, as its power consumption can feasibly be brought under a few dozens of micro-watts on a not so modern CMOS process (such as a 0.5 μm commercial process), in contrast with any commercial micro-controller-

[1] Application Specific Integrated Circuit

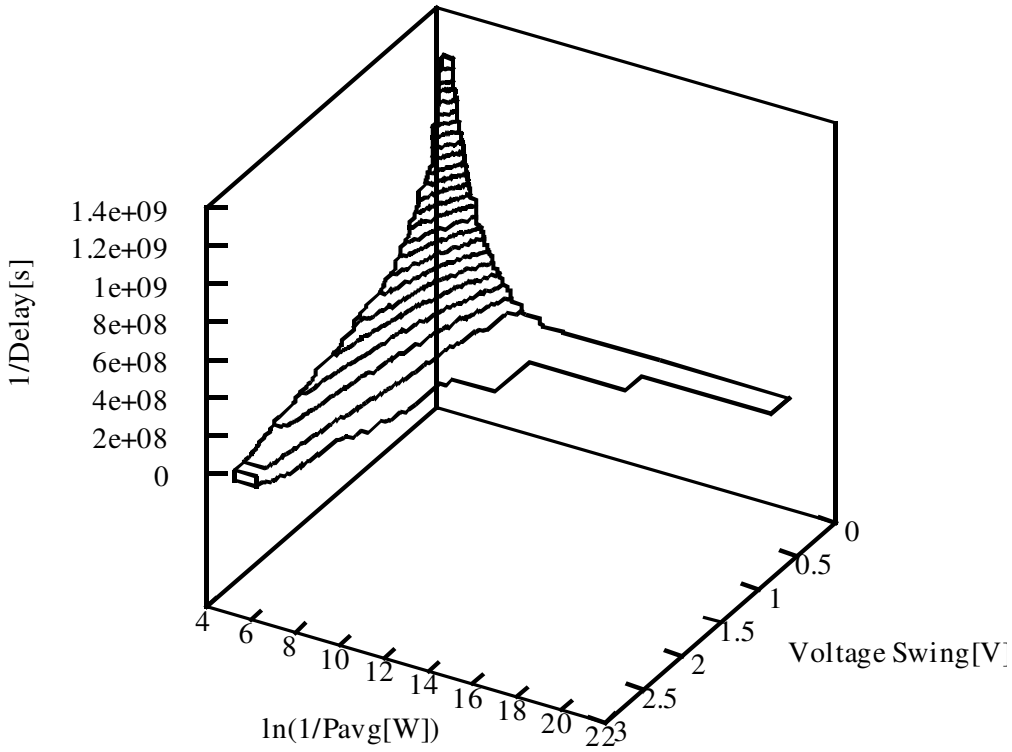


Figure 3.3: Xor Pareto front

based implementation or even a FPGA[2] one, that easily surpass such limits.

Three phases characterize each node's operation: impulsive sound detection, sound classification and a stage of spacial location of sounds.

For the detection stage, a series of hardware algorithms were evaluated in order to determine the optima in terms on low power consumption and detection efficiency [10], and a hand-made first version of the resulting integrated circuit has been designed and tested, as shown in [9]. The circuit implements a continuous-time wavelet transform (CWT), as a simple way of performing signal detection and classification. Fig. 3.9 presents the detection stage, which is composed by a Gm-C filter bank that separates the input signal into three CWT coefficients whose maximum frequency is 875 Hz. Then an energy estimation is performed for each coefficient, their sum is computed and compared to an adaptive threshold, typically a running average or a RMS estimation of the same pre-processed signal, scaled by a gain factor.

The first problem at hand is to optimize the OTAs (Operational Transconductance Amplifiers) such that its power and parasitics are minimized, whereas slew rate, frequency response and linearity are maximized. This entails an more efficient filter, closer to the theoretical specifications given in [10], and that fixes the problems of its first version, as given in [9]: pole-shifting, excessive DC-offset and higher than expected power consumption due to the required adjustment of the filter's poles (5.64 μA from the expected 2.26 μA of total bias current in the filter bank alone, due to the needed doubling of the OTAs' bias (from 45 nA to

[2]Field Programmable Gate Array

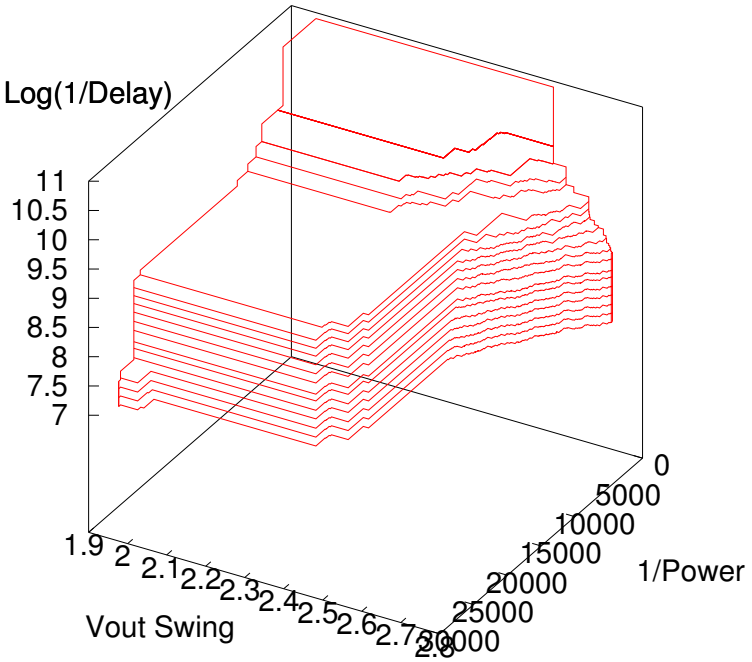


Figure 3.4: A Pareto front for a MCML Nand gate

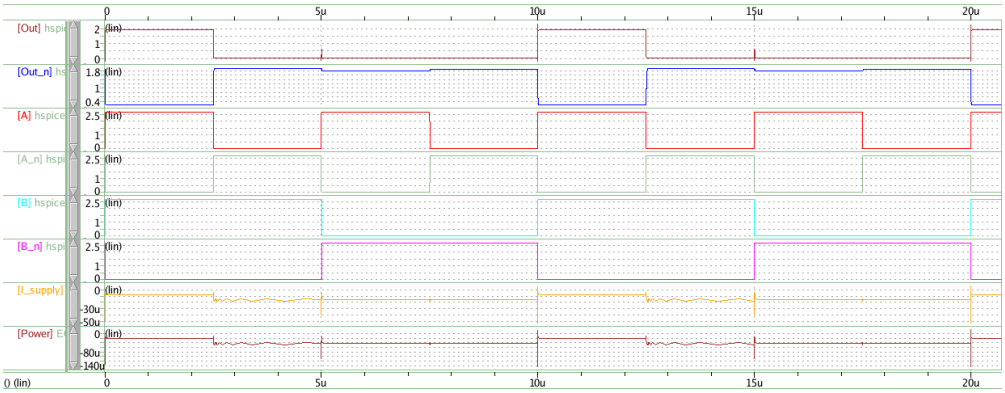


Figure 3.5: Timing simulation for the MCML Nand gate

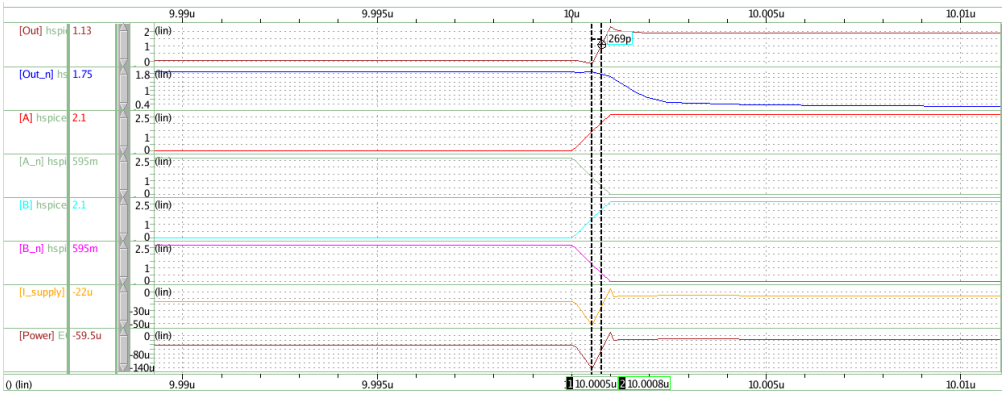


Figure 3.6: Delay measuremt in the simulation diagram

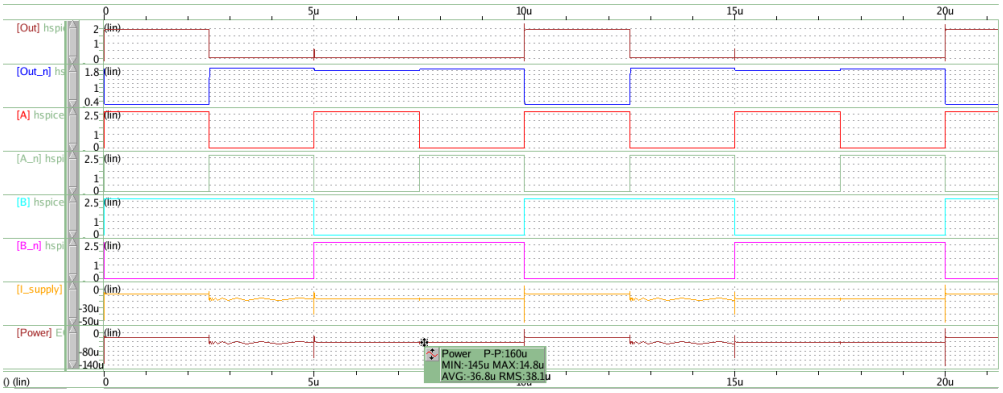


Figure 3.7: Average Power curve

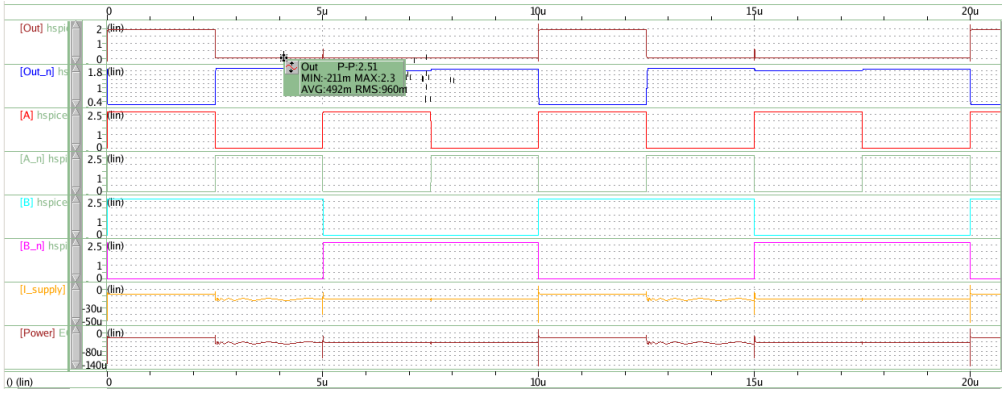
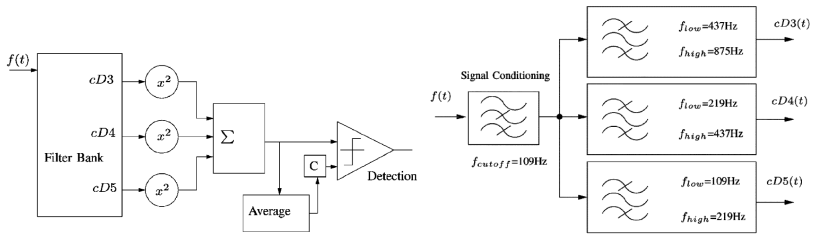


Figure 3.8: Voltage swing waveform



(a) Architecture of the detector. (b) Filter bank equivalent.

Figure 3.9: Architecture of filter bank and the detector, as based on the structure proposed in[10]. The filter is implemented Gm-C circuits, which entails the need for low power, highly linear OTAs.

95.6 nA) in order to place the poles in the right frequencies). Though some of the problems in the first version are due to the final circuit's layout parasitics and uncertainties unaccountable during the simulation process, it is clear that the minimization of the systematic defects in the design, should greatly compensate for post-fabrication random effects (this of course also opens an interesting problem for the CAD future development: how to integrate CMOS fabrication uncertainty models into it in order to account for such deviations during the optimization cycle, and minimize if possible their impact in the final design). As it was already mentioned, we envision that the Pelgrom model can be used as a way to determine transistor size constraints and feed them into the tool such that more robust circuits can be obtained.

3.2.1 Optimization of an Operational Transconductance Amplifier

As shown in[48], an operational transconductance amplifier is optimized in order to solve the problems of the CWT filter's first implementation. See Fig. 3.10.

The output transconductance G_m can be approximated by:

$$G_m = \frac{g_{m1}}{m(1 + \frac{g_{m1}}{4g_{m2}})}, \tag{3.4}$$

where m represents the scale factor due to the lower current mirror, g_{m1} and g_{m2} represent the transconductance of transistors M_1 and M_2, respectively.

OTAs are specially characterized by their linear range of transconductance, defined as the differential input voltage range that produces a constant value of transconductance at the output. Since OTAs are not perfectly linear circuits, their transconductance can vary depending on the amplifier's design and operating mode. Consequently, the designer must define the precision of its linear range, based on the variability than can be tolerated, distortion or the maximum deviation in the circuit's response. Most experiments and simulations executed show that the linear range ΔV is directly proportional to the bias current in DC and to the dimensions of transistors M_1 and M_2:

$$\Delta V = f_{(I_b, W_1, L_1, W_2, L_2)} \tag{3.5}$$

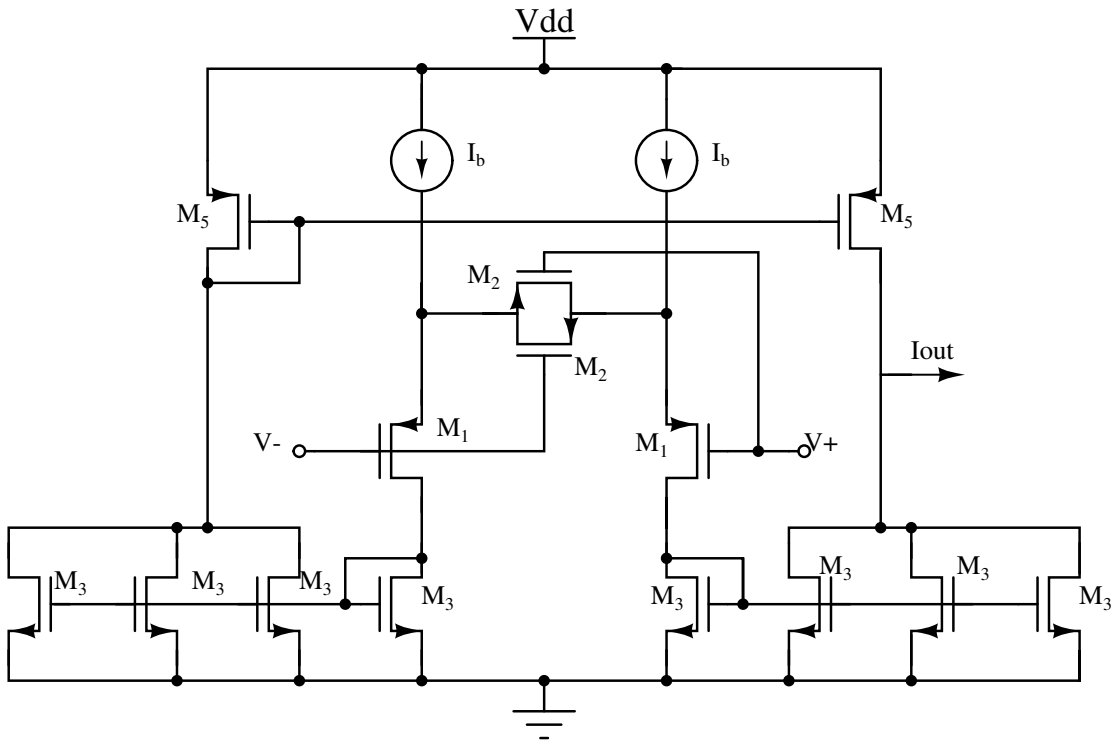


Figure 3.10: Esquemático del *OTA* de 192nS.

$$SR = \frac{2 \cdot I_b}{m \cdot C} \tag{3.6}$$

$$SR = \frac{2 \cdot I_b \cdot (2\pi f_c)}{m \cdot G_m}, \tag{3.7}$$

Equations (3.4), (3.5), (3.6) and (3.7) show competing objectives, i. e. conflicting fitness functions. This means that it is not possible to find an single optimum for a design metric (either ΔV, SR or G_m) without negatively affecting the other design metrics or fitness values.

The concepts related to optimization, genetic algorithms and the Pareto Front were detailed in Chapter 2. Building upon those concepts, Fig. 3.11 shows the three-dimensional Pareto front of the previously shown OTA. This front contains 1500 individuals (parametrizations) and it was generated by the PESA genetic algorithm. The graphic shows the design trade offs between the three parameters: in order to maximize one of these, the other parameters must decrease their values.

Table 3.3 contains a list of some selected results given by the optimization tool. These results contain cases with wide linear range, high transconductances, and/or low capacitances. The slew rate condition is fulfilled in low G_m values.

Figs. 3.12 and 3.13 present the simulation results for the best overall case found, according to the requirements presented early. The first figure shows the output current response for an input voltage sweep from -1V to +1V. The second figure presents the transconductance

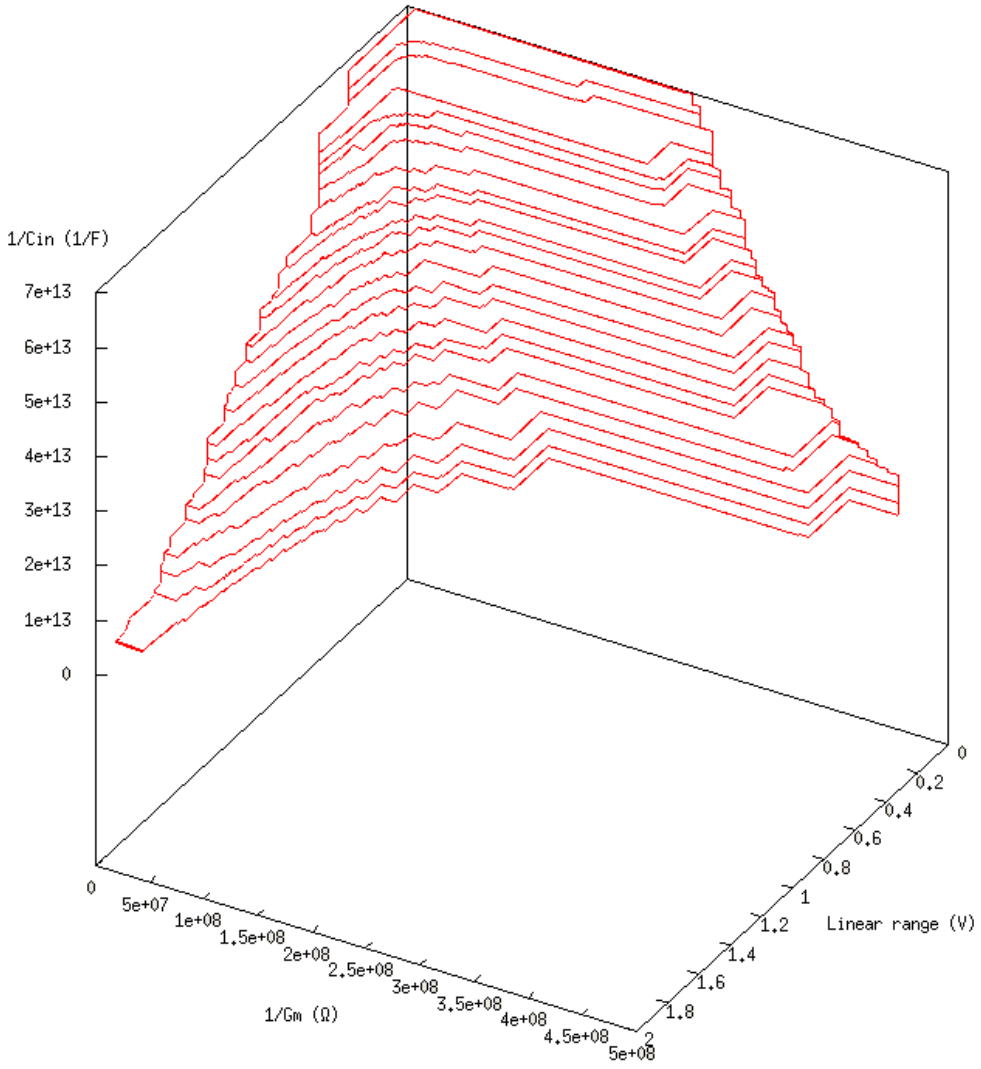


Figure 3.11: Pareto front of the designed OTA. The graphic contains three metrics: input capacitance, transconductance and linear voltage range.

Table 3.3: Extraction of some representative results given by the optimization tool

G_m (nS)	ΔV (mV)	C_{in}(fF)	Slew rate (mV/μs)	I_b(nA)
33.55	$\pm$949	235.58	2.131	87.226
40.90	$\pm$581	86.35	1.748	64.871
44.80	$\pm$624	93.40	2.054	52.097
28.90	$\pm$549	94.63	2.473	39.323
18.0	$\pm$504	85.74	3.971	26.548
16.55	$\pm$506	88.64	4.319	20.161

of the circuit, which is obtained by calculating the slope of Fig. 3.12.

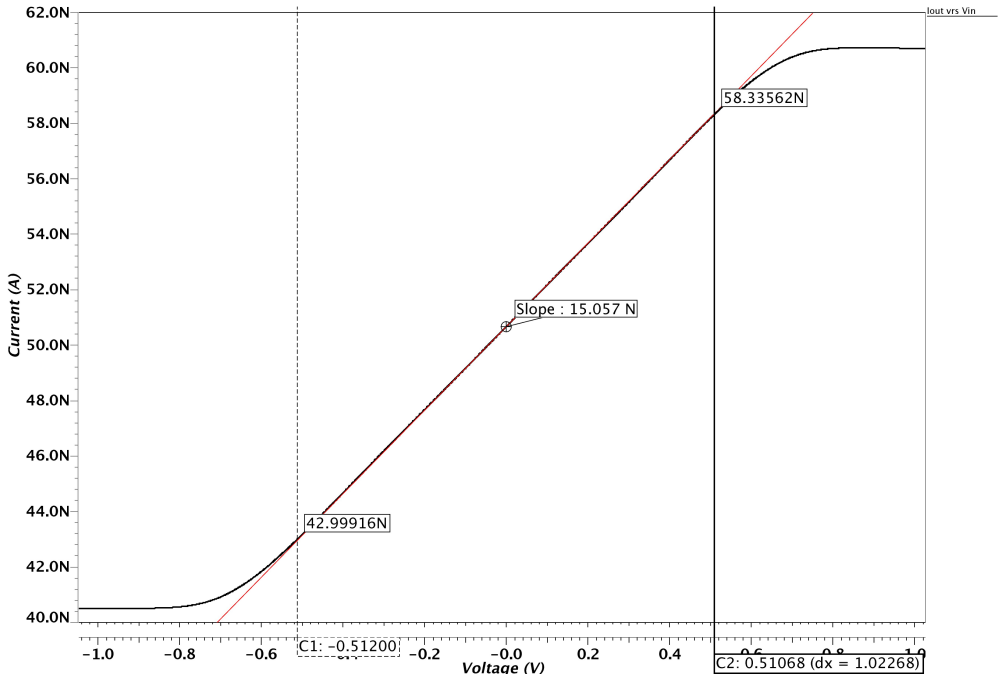


Figure 3.12: Output current curve as a function of the input voltage. The slope of this curve gives the circuit transconductance. All the waveforms presented are obtained with Mentor Graphics Eldo simulator and EzWave viewer.

Table 3.4 shows a summary of the simulations obtained for the best case OTA and the initial OTA, that shows the improvements obtained in comparison with the original design. The decrease of the transconductance value is a positive result because in order to keep the pole of the filter in the same place, when the transconductance is reduced it is necessary to reduce the capacitance too, producing a reduction of the circuit area. All of the other design specifications where also greatly improved: the input capacitance and power consumption were lowered while the linear range and the slew rate were greatly improved. Finally, Table 3.5 contains the unitary transistor dimensions given by the optimization tool.

Table 3.4: Simulation results for the best case OTA

Measurement	Best case OTA	Initial OTA
Maximum G_m (nS)	15.077	36.57
Linear range ΔV (mV)	$\pm$512	$\pm$260
Slew rate (mV/μs)	3.676	1.954
Power consumption (nW)	144.3	174.93
Input capacitance (fF)	89.63	267.79

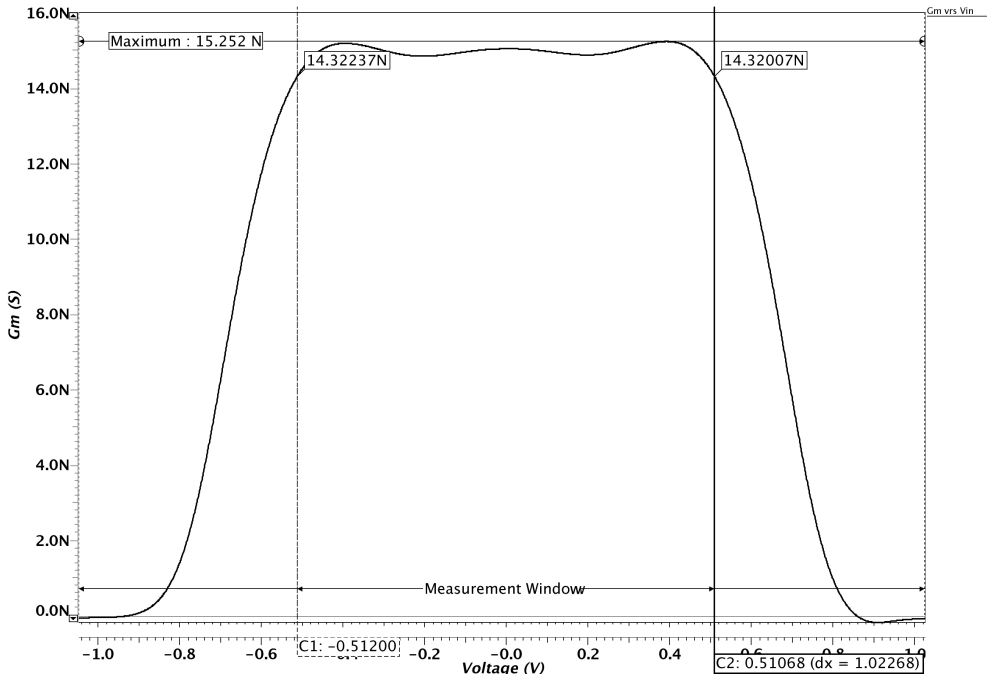


Figure 3.13: G_m curve as a function of the input voltage for the selected result

Table 3.5: Transistor dimensions for the best case OTA

Parameter	Value
L_1 (μm)	7.2
L_2 (μm)	8
L_3 (μm)	0.5
L_5 (μm)	5.4
W_1 (μm)	0.8
W_2 (μm)	0.8
W_3 (μm)	7.2
W_5 (μm)	7.2

3.2.2 Optimization of the Gm-C filters

As discussed in [48], an operational transconductance amplifier has been optimized in order to minimize some of the the problems of the CWT filter's first implementation. The use of a automatized tool for multiple objective design based on Genetic Algorithms made easier to obtain low transconductance, low power OTAs with a linear range over 1V. The complete filter is shown in Fig.3.14. A third order pass-band filter is used to obtain each coefficient. Some extra amplification is given at the input, and a high-pass filter gets rid of any DC offset from previous stages. This filter introduces an extra pole that is not considered in the theroretical relations specified in [10]. Its effect is minimized by placing it at a very low frequency, and it only impacts the cD5 coefficient (see Fig. 3.15 for the simulated frequency response of the whole CWT filter). This effect should not nonetheless present major deviations for the expected detection efficiency, as explained in [10]. was discussed in [10], this coefficient is only determinant in the minimization of false positives.

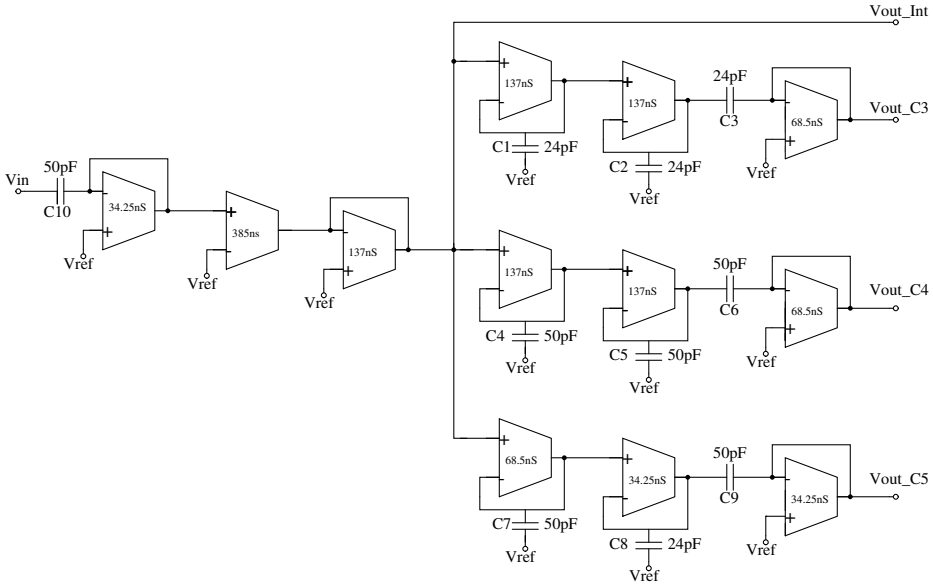


Figure 3.14: Gm-C implementation of the CWT.

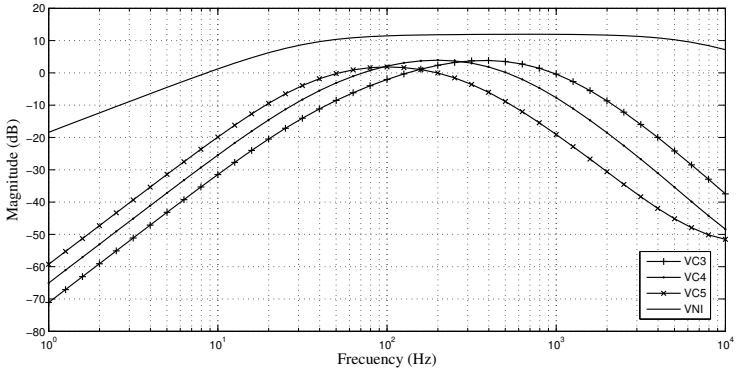


Figure 3.15: Post-layout magnitude frequency response of the Gm-C filter bank. Vint is the response for the intermediate node, after the input amplifier.

Filter construction.

A full description of the process carried out in order to get rid of the problems of the first implementation, as reported in [9] is given. This implementation presented excesive power consumption, large parsitic capacitances and frequency pole-shifting in the filter bank. In the optimized implementation the structure of the filter was not touched but all components were resized in order to reduce the aforementioned problems.

A natural question would be why not to optimize the whole filter structure instead of dividing it into blocks and individually optimizing each one? There is no single answer to this question but hierarchical design is well-established, simulation time grows with circuit complexity and, since the optimization strategy relies on thousands of simulations in order to achieve the evolutionary process, see section 2.2.1, the total computing time would explode to prohibitive levels. Other optimization techniques were discussed in section 2.1, although those techniques also work at the block level.

The first step was to use a 192nS OTA, taking advandage of the technique employed in [43] and [3], where current mirrors with a current scaling factor of m=0.3 are used in order to obtain such a transconductance. To achieve this current scaling, a 64nS OTA like shown in Fig. 5.3 was modified. Table 3.6 shows the corresponding transistor dimensions.

Table 3.6: Unitary transistor dimensions for the OTAs.

Transistor	W(μm)	L(μm)
M_1	0.8	7.2
M_2	0.8	8
M_3	5	2
M_5	5	2

It must be pointed out that these trasistor dimensions are of unitary transistors, which are associated to form the transistors shown in Fig. 5.3. Thus, transistors M_1 in Fig. 5.3, are formed by the series association of 3 unitary transistors, the same applies to M_3 and M_5. M_2 is formed by series of 18 transistors.

Fig. 3.10, in section 3.2.1 shows the 192nS OTA, with the current-scaling transistors added to the circuit. The modifications were made in the lower transistors, those named as M_3 in the figure. The bias current for this circuit is 20.16nA.

Current, transconductance and transient responses.

The following figures show the results from schematic simulation of the 192nS OTA. Fig. 3.17 shows the output current as a function a differential input voltage of $\pm$ 1V.

Since the transconductance is the slope of 3.17, its first derivative was computed and ploted in Fig. 3.18. As can be seen in the figure, a maximum transconductance of 189.89nS was reached, which is a 1.09% deviation from the theoretical value.

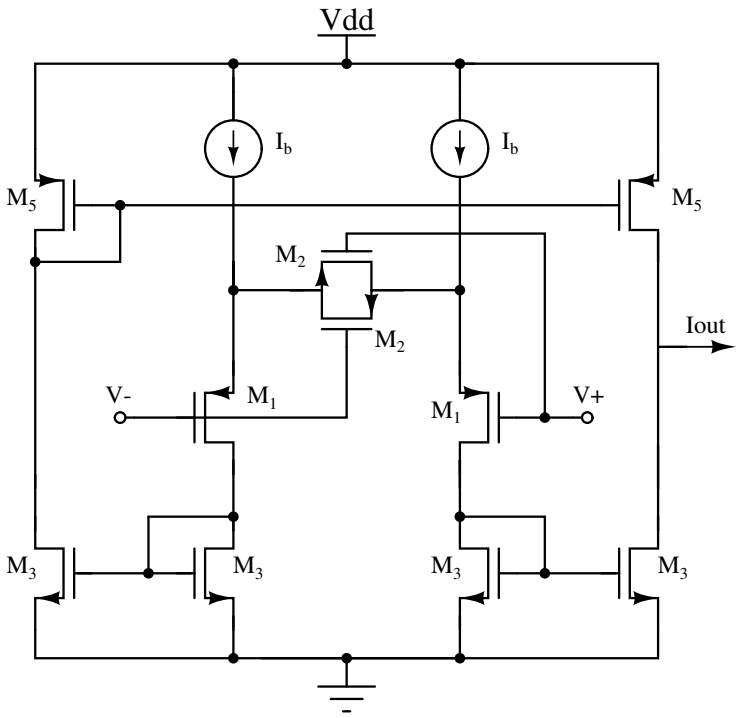


Figure 3.16: Circuit schematic for a 64nS OTA.

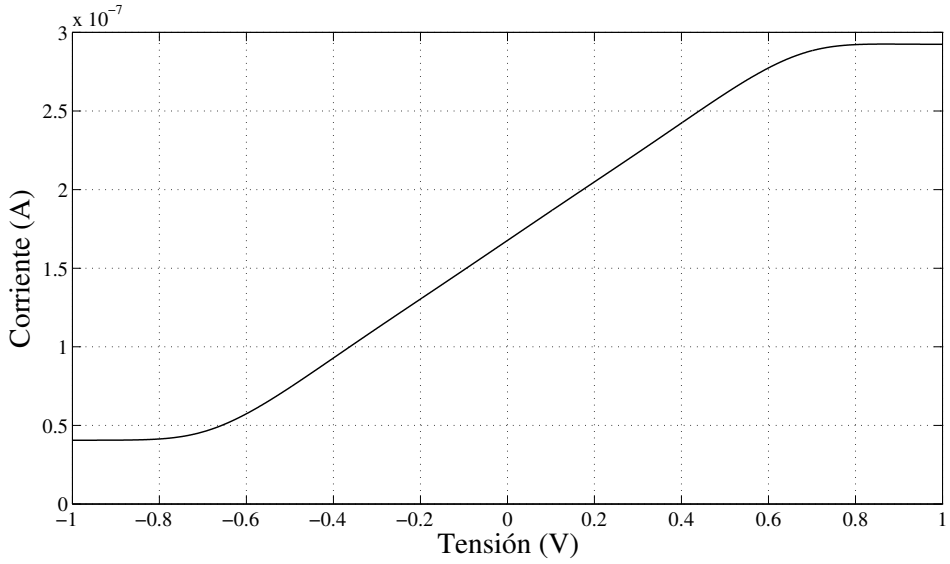


Figure 3.17: i_{out} vrs V_d for the 192nS OTA.

Fig. 3.19 shows the behavior of the transconductor's output current, when an AC voltage is applied to its inputs. Notice how the output current saturates due to the input voltage range reaching the OTA's non linear region, where its transconductance drops down sharply.

Table 3.7 summarizes the main features of interest for the 192 nS transconductor. These data show that current scaling does not affect the linear range nor the input capacitance, when compared to other OTA designs reported in [43].

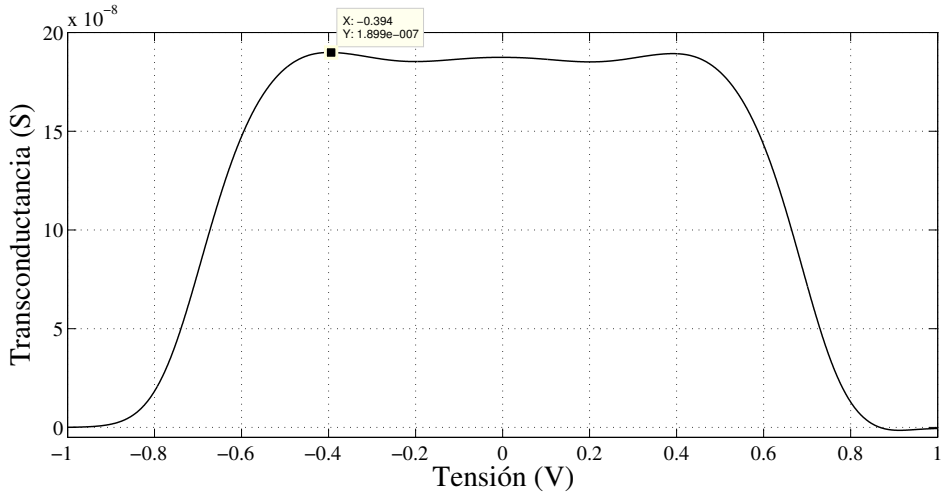


Figure 3.18: Transconductance curve of the 192nS OTA.

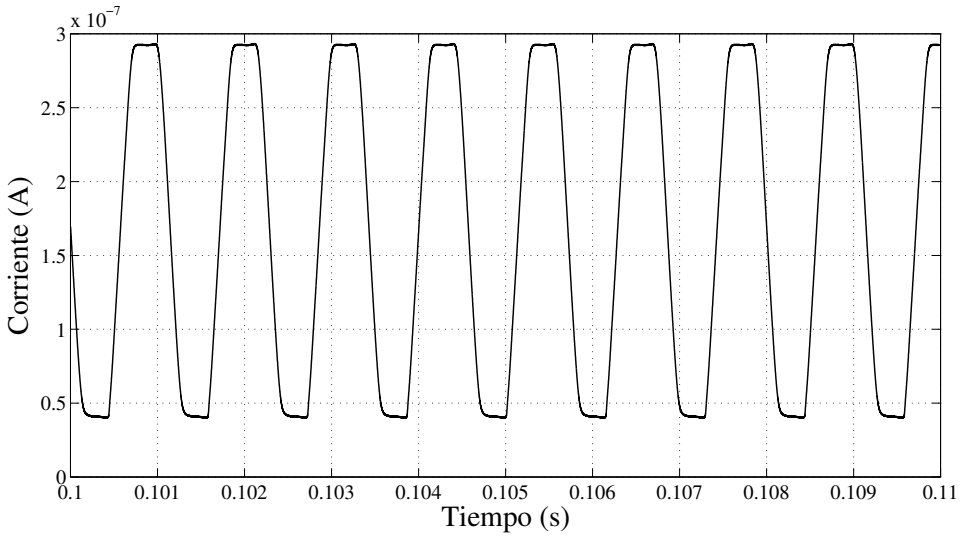


Figure 3.19: Output current transient response for the 192nS OTA.

Analog filter bank.

The original filter is shown in Fig. 3.14, in order to optimize its behavior the following changes were made: substitution of all of its OTAs and also its capacitances. The 385nS was changed by 192nS, and the 137nS, 68.5nS and 34.25 were changed by 64nS, 32nS and 16nS OTAs, respectively. Recalculation of capacitances was necesary as the filter bank had to maintain its cutoff frequencies. The required frequency response is the one shown in Fig. **??**. Equation (3.8) was used to obtain these new capacitance values, where G_m is the

Table 3.7: Performance features of the 192nS OTA.

OTA	G_{mmax}(nS)	C_{in} (fF)	Power (nW)	ΔV(mV)	SR (mV/μs)
192	189.89	89.63	914.54	$\pm$502	0.485

transconductance of the OTA associated to the capacitor and f_c the corresponding cutoff frequency.

$$C = \frac{G_m}{2\pi f_c} \tag{3.8}$$

Table 3.8 shows the capacitance values for the redesign filter. Shown cutoff frequencies are of every first order filter which conform each filter band. Every coefficient is generated by two first-order low pass filters and a first-order high pass filter at the output. The input filter is a first-order high pass followed by an amplification stage.

Table 3.8: Associated capacitance and G_m values in the filter bank.

Capacitor	f_c(Hz)	G_m (nS)	Capacitance (pF)
C1	875	64	12
C2	875	64	12
C3	437	32	12
C4	437	64	24
C5	437	64	24
C6	219	32	24
C7	219	32	24
C8	219	16	12
C9	109	16	24
C10	109	16	24

Optimized filter bank

Having the optimized OTAs and the new capacitance values, the filter structure shown in 3.20 was reimplemented.

The transfer function for each of the filter's output coefficients is calculated. Coefficient 3 is shown in equation (3.9). This function has a second-order pole at 424Hz and a first-order pole at 845Hz.

$$CD_3(s) = \frac{sC3(64nS)^2}{(sC1 + 64nS)^2(sC3 + 32nS)} \tag{3.9}$$

Equation (3.10) shows coefficient 4 with poles at 424Hz and 212Hz.

$$CD_4(s) = \frac{sC6(64nS)^2}{(sC4 + 64nS)^2(sC6 + 32nS)} \tag{3.10}$$

Equation (3.11) corresponds to coefficient 5, with its poles at 106Hz and 212Hz.

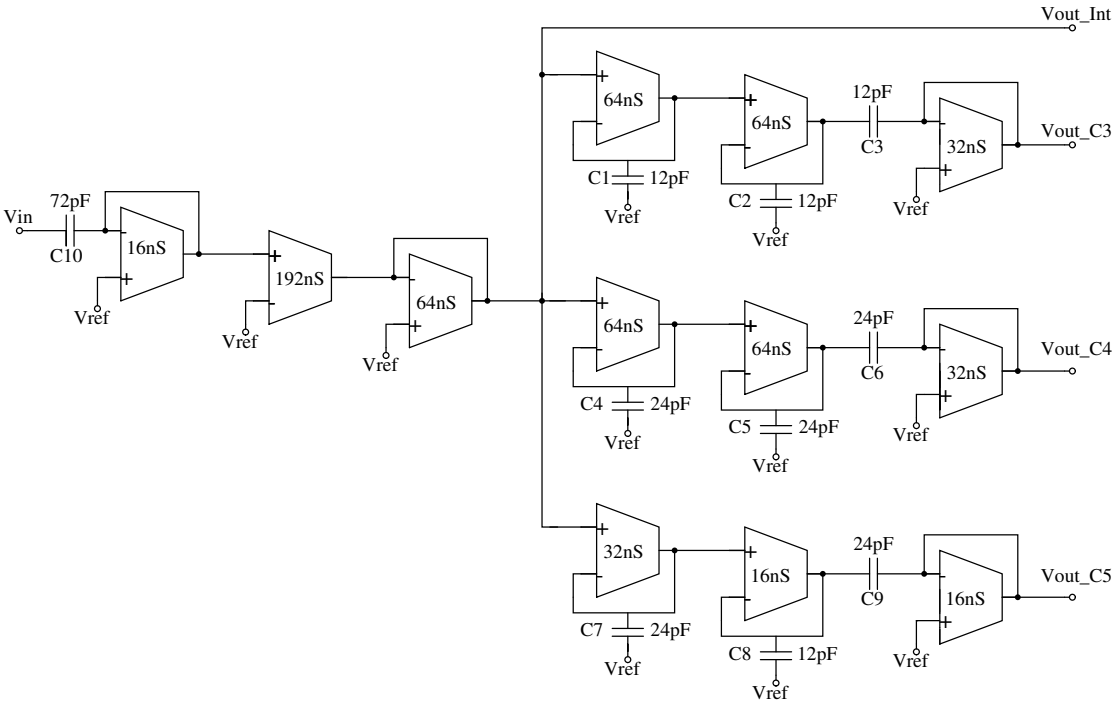


Figure 3.20: Optimized filter bank diagram, where the OTA stages correspond to the optimization discussed in section 3.2.1.

$$CD_5(s) = \frac{sC9(16nS)(32nS)}{(sC7 + 32nS)(sC8 + 16nS)(sC9 + 16nS)} \tag{3.11}$$

Equation (3.12) shows the transfer function for the intemediate node.

$$Vout_{NI}(s) = \frac{sC10(\frac{192nS}{64nS})}{sC10 + 16nS} \tag{3.12}$$

Figs. 3.21 and 3.22 show the magnitud and phase frequency responses, respectively, both obtained from the theoretical transfer function.

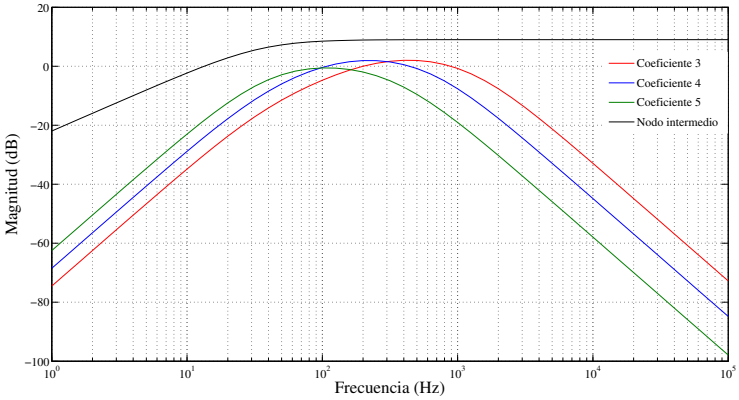


Figure 3.21: Theoretical magnitude frequency response.

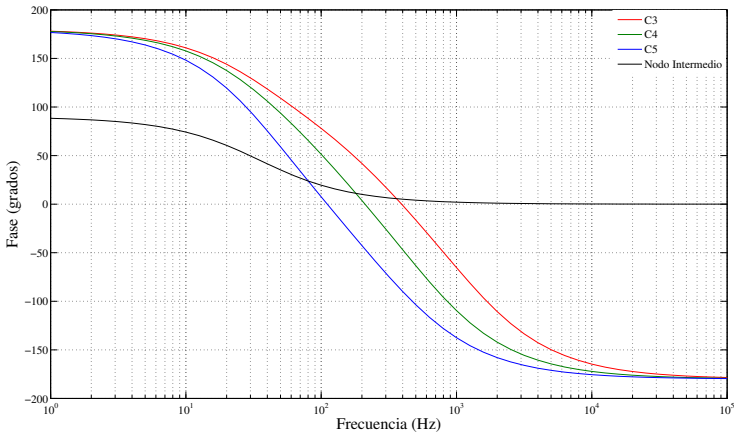


Figure 3.22: Theoretical phase frequency response.

Biasing current mirrors

Every OTA in the filter works at a bias current of 20,16nA. Fig. 3.23 shows an instance of the current mirrors used to bias the whole filter, transistors M1 and M2 have the same size and their dimensions are: L=20μm and W=3μm.

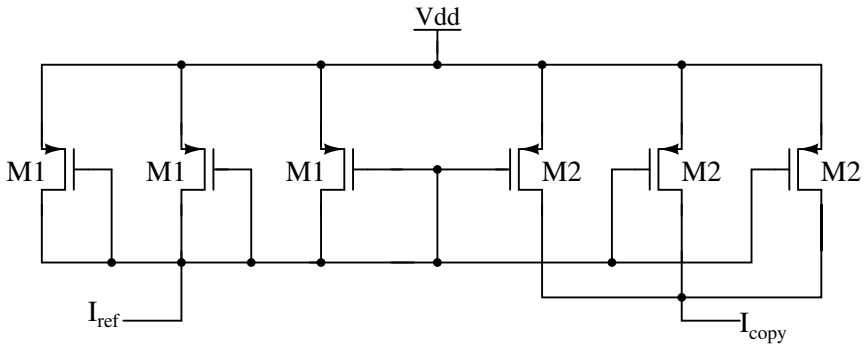
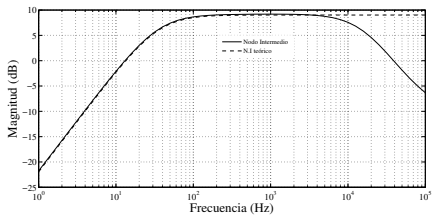


Figure 3.23: Current mirror biasing circuitry.

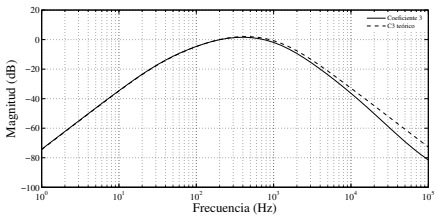
Schematic simulation results.

After the initial theoretical análisis, schematic driven simulations were carried out in order to assess the behavior of the optimized circuit. Fig. 3.24 shows the magnitude frequency response and Fig. 3.25 the corresponding transient response. Fig. 3.26 also shows a transient response but for several frequencies of the input signal.

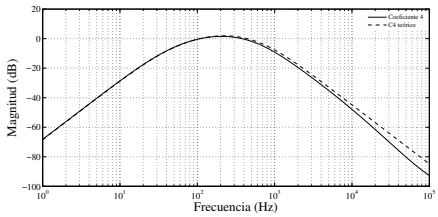
Looking at the magnitude values in the frequency response, there is a slight difference between the maximum value of the coefficients. In Fig. 3.24, these differences have been reduced, specially between coefficients 3 and 4, but not so much for coefficient 5. When adding the input filter in order to get rid of DC effects, it has an effect on the polinomial, son rising the value of C10 moves the corresponding pole to the left and reduces the negative effect on the magnitude, at least for coefficients 3 and 4. Although this effect remains on coefficient 5, it was proved in [10] that this coefficient is less determinative than the other two. The results obtained when changing C10 are presented in 3.9.



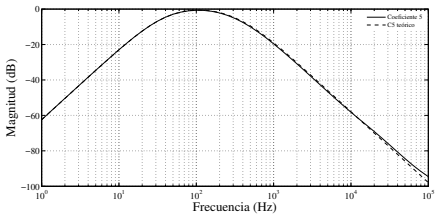
(a) Frequency response for the intermediate node



(b) Frequency response for coefficient 3.



(c) Frequency response for coefficient 4.



(d) Frequency response for coefficient 5.

Figure 3.24: Magnitude frequency response for coefficients 3, 4, 5 and intermediate node.

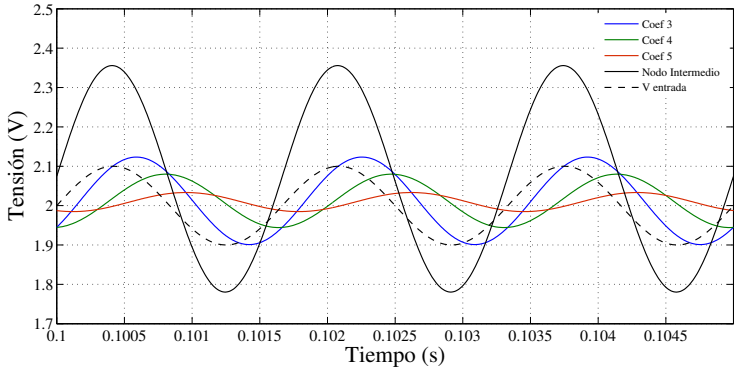
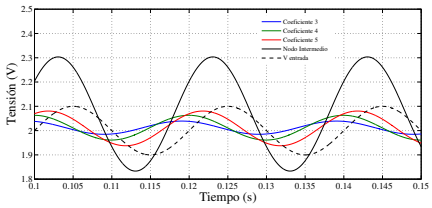


Figure 3.25: Filter transient response for an input of 600Hz of frequency.

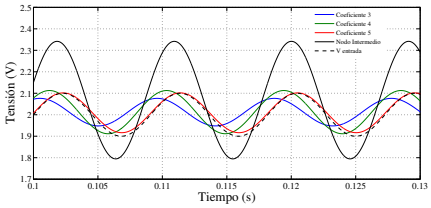
As can be seen in table 3.10 the intermediate's node capacitance was significantly reduced – a 90% – when compared to the original implementation reported in [9]. This helps in reducing the pole-shifting problem. As can be seen in Fig. 3.24, the highest value poles are

Table 3.9: Frequency response magnitude variations for each coefficient due to change in C10.

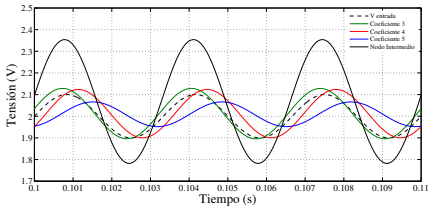
C10 (pF)	Coef. 3 (dB)	Coef. 4 (dB)	Coef. 5 (dB)	Interm. Node (dB)
24	1.19	0.68	-2.71	9.08
48	1.49	1.31	-1.16	9.15
72	1.54	1.48	-0.73	9.19
96	1.59	1.57	-0.49	9.20



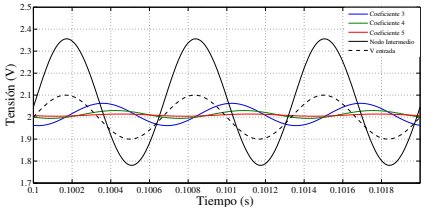
(a) Transient response to voltage input at 50Hz.



(b) Transient response to voltage input at 110Hz.



(c) Transient response to voltage input at 300Hz.



(d) Transient response to voltage input at 1500Hz.

Figure 3.26: Filter transient behavior for 100mV peak input at several frequencies.

shifted to the right due to the reduction of the size of the OTAs, which directly impact the gate capacitance. This capacitance is the main contributor to the intermediate node total capacitance. There are, however, differences in the higher frequency regions between theoretical and schematic-driven simulation frequency responses. Nevertheless, in these regions the atenuation is so high that there is no effect on the desired behavior of the filter.

After optimization of the OTAs, the power consumption was 3.66μW, which represents a 34.29% reduction with regards to [9], without the need for current adjustments. See table 3.11.

Table 3.10: Filter's obtained features vs initial ones.

Feature	Initial	Obtained values	Improvement (%)
Power (μW)	5.57	3.66	34.29
Intermediate node capacitance (pF)	11.21	1.03	90.77

Table 3.11: Filter bank's current consumption at 4V. IC implemented in [9].

Circuit	Consumption with no adjustment (μA)	Consumption with adjustment (μA)
Filter Bank	2.26	5.64

Table 3.12 presents the *offset* values at the intermediate node and coefficient outputs.

Table 3.12: Output *offset* voltage for the filter bank.

Stage	*offset* Voltage (mV)
Coeficient 3	12.06
Coeficient 4	12.06
Coeficient 5	9.04
Intermediate node	68.06

Post layout filter results

Post-layout simulation results for the bank give a total power consumption of 7μW 4VV_{DD} supply, a third of what was reported in [9], but including 2μW dissipated on the three OpAmps connected as followers and placed at the output of each coefficient. These OpAmps (not shown in Fig. 3.20) are there only to minimize the effect of the capacitance of the pads in the filters' response. Since these pads are mere measurement test points for this prototype, they would not be required in a final implementation, which means that the real power consumption ot the circuit should be around 5μW. The final area of the circuit is of 1.716 mm^2. Fig. 3.15 shows the obtained frequency response.

3.2.3 Optimization of the computing unit.

The energy computing unit can be seen at the right-hand side in Fig. 3.9. It produces the sum of the rectified three coefficients computed by the filter (in this case, coefficients 3, 4 and 5, as shown in Fig. 3.29). In this case the goal is to have a detection stage with the lowest power consumption possible, while minimizing DC systematic offset that does not allow for a perfectly symmetrical rectification of each coefficient. This error could cause false gunshot detection (false alarms) which must be avoided. False detection not only lowers the detection efficiency, but causes wasted energy (a very limited resource) since it may imply the waking up of subsequent units of the WSN that verify the likelihood of the alarm. The optimized OTAs from the previous section will be reused here, such that the optimization problem is limited to other sections of the circuit.

Design of a two-stage comparator.

Fig. 3.27 shows a two-stage comparator that combines the features of the differential amplifier with the qualities of an inverter stage. The poor gain of the differential stage is increased by the gain of the inverter stage. The output of the differential stage, which is about V_{DD}, is in the vicinity of the transition point of the inverter stage that follows. Therefore, the limited output range, which is a problem for the differential stage, now is a good feature in the two-stage configuration.

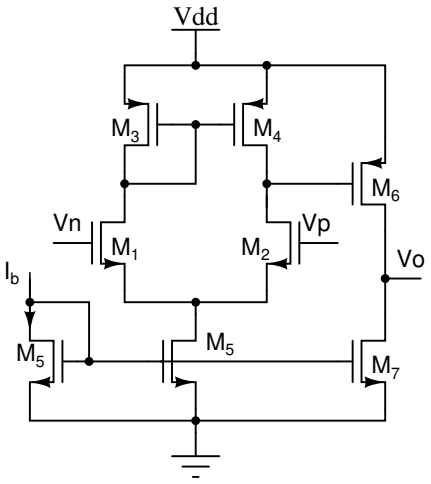


Figure 3.27: Schematic diagram for a two-stage comparator.

Comparator's initial design and its implementation with an automated optimization tool.

As already stated, analog circuit dimensioning has the particularity that by varying some of its parameters will affect another of its characteristics. For example, when trying to reduce the power consumption, the systematic *offset* or the duty cycle of the comparator can be adversely affected. This strong bond between parameters and performance makes it necessary to iterate in the design process to find an optimal compromise between the circuit's requirements.

For comparison the initial design parameters of the comparator are used. These were obtained with hand made simulations, which are shown in Table 3.13. This step is not stricly necessary although it might help to have an acceptable starting point for the optimization process. Therefore, it was decided to try it that way. What it does matter is to have good circuit parameters' constraints such that meaningful results can be obtained. A full discusion of a sizing rules methodology is given by Graeb in [27].

Table 3.13: Initial parameters for the comparator, with a bias current of $20\mu A$

Transistor	**Dimensions** W/L (μm)
M1	20/10
M2	20/10
M3	10/10
M4	40/10
M5	10/10

Optimization tool modifications

The main changes are related to the computation of the *fitness values*. These are shown in table 3.14, the *slew rate* is determined as:

$$SR_{(i)} = dV_{out(i)} = \frac{V_{out(i+1)} - V_{out(i-1)}}{2d} \tag{3.13}$$

El valor de corriente de polarización y el consumo de potencia se obtienen directamente del archivo de simulación. Finalmente, el *offset* se obtiene encontrando el valor de la tensión de entrada del comportamiento en CD para el cual la tensión de salida es igual a la referencia (en este caso $VDD/2$) y restándoselo a la referencia.

where i is the position in which the derivative is evaluated, $V_{out(i-1)}$ and $V_{out(i+1)}$ are the voltage values before and after the evaluation, d is the time between two samples, $10\mu V$ as configured in the simulation environment is. The maximum rate of change from low to high and from high to low is obtained in order to choose the smallest value of both (the smallest being the worst case) as the corresponging *slew rate*.

The bias current value and the power consumption are obtained directly from the simulation output file. Finally, the *offset* is obtained by finding the value of the input voltage in the DC behavior in which the output voltage is equal to the reference (in this case $VDD/2$) and then subtracting it from the reference.

Table 3.14: Comparator's fitness values.

Specification	**Fitness Value**
slew rate	$\frac{dV_{out}}{dt}$
Ib	$\frac{1}{Ib}$
consumo	$\frac{1}{consumo}$
offset	$\frac{1}{offset}$

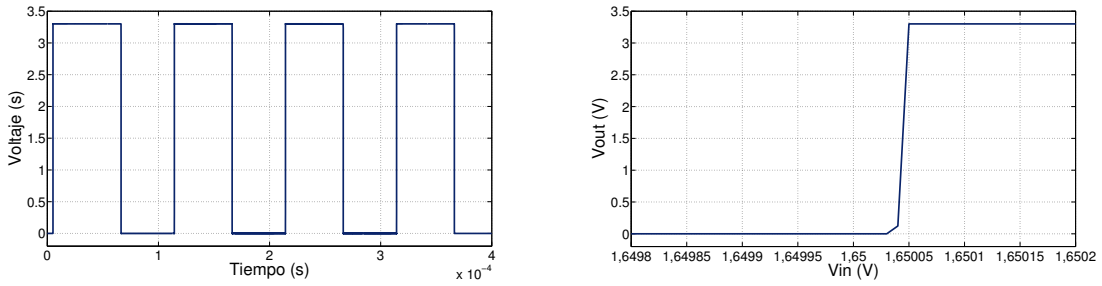
Luego de los cambios realizados se simuló de nuevo el comparador, obteniéndose el comportamiento de la figura 3.28 donde se aprecia la simetría del ciclo de trabajo, manteniéndose el *offset* sistemático por debajo de los $50\mu V$ con un consumo de potencia similar a lo obtenido de la simulación en esquemático. El resumen de las características *post-layout* se muestra en la tabla 3.15.

After these changes the comparator was simulated yielding the behavior of Fig 3.28, where the symmetry of the duty cycle can be appreciated, at the same time keeping the systematic *offset* below 50μ V with a power consumption similar to that obtained from the schematic simulation. Table 3.15 presents a summary of the features from *post-layout* simulation.

Table 3.15: Comparator's post*layout* characteristics.

Offset (μV)	**Duty cycle** (%)	**Ib** (nA)	**Consumption** (nA) @$3,3V$	***slew rate*** $V/\mu s$
35	49,9	14,5	35,4	401

The unit fabricated in[9] has the response shown in Fig. 3.31. The rectifier has a asymmetric response, which is precisely a consequence of the asymmetry in the comparator's switching



(a) Transient response

(b) DC response $Vout$ versus Vin.

Figure 3.28: Comparator's *post-layout* simulation for a sine wave input of $0,15V$ at $10kHz$.

voltages or tripping point, and that of the associated OTA. This asymmetry is up to 7% of the input signal. Current consumptions of the first version are shown in table 3.16 for two different supply voltages.

Table 3.16: Bias and current consumption for circuit implemented in[9].

Supply (V)	**Bias (nA)**	**Consumption (μA)**
3.3	90.15	1.813
4.0	95.03	6.800

One can minimize random offset by using correct circuit layout techniques (but as stated in the previous case, an improved version of the CAD tool, that takes into account this effects during the optimization process, should be the next development step). But systematic offset can be already minimized through simulations, along with power consumption and other metrics.

Table 3.17: Final characteristics of the optimized comparator for the rectifier.

Systematic offset (μV)	Comparator's duty cycle (%)	Input bias (nA)	Slew rate (V/μ)
25	49.05	14.5	411,5

Optimized OTAs from Section 3.2.2 where used to complete the unit. Post-layout simulation results from the whole rectifier circuit are shown in Fig. 3.32, for a 850Hz sine input signal, which shows an almost negligible systematic offset error (548μV). There is a small distortion in signal's zero crossing, that nonetheless is significantly lower that in the simulations from the hand-made version. Final power consumption in the unit was cut from 27 μW in the original design to only 2.54 μW for a 4V V_{DD} supply.

3.3 Optimization of a Self-biased Current Source

In the first version of the detection circuit, a standard self-biased current source (SBCS) was used. This of course implied extra wasted power (in the order of hundreds of microwatts)

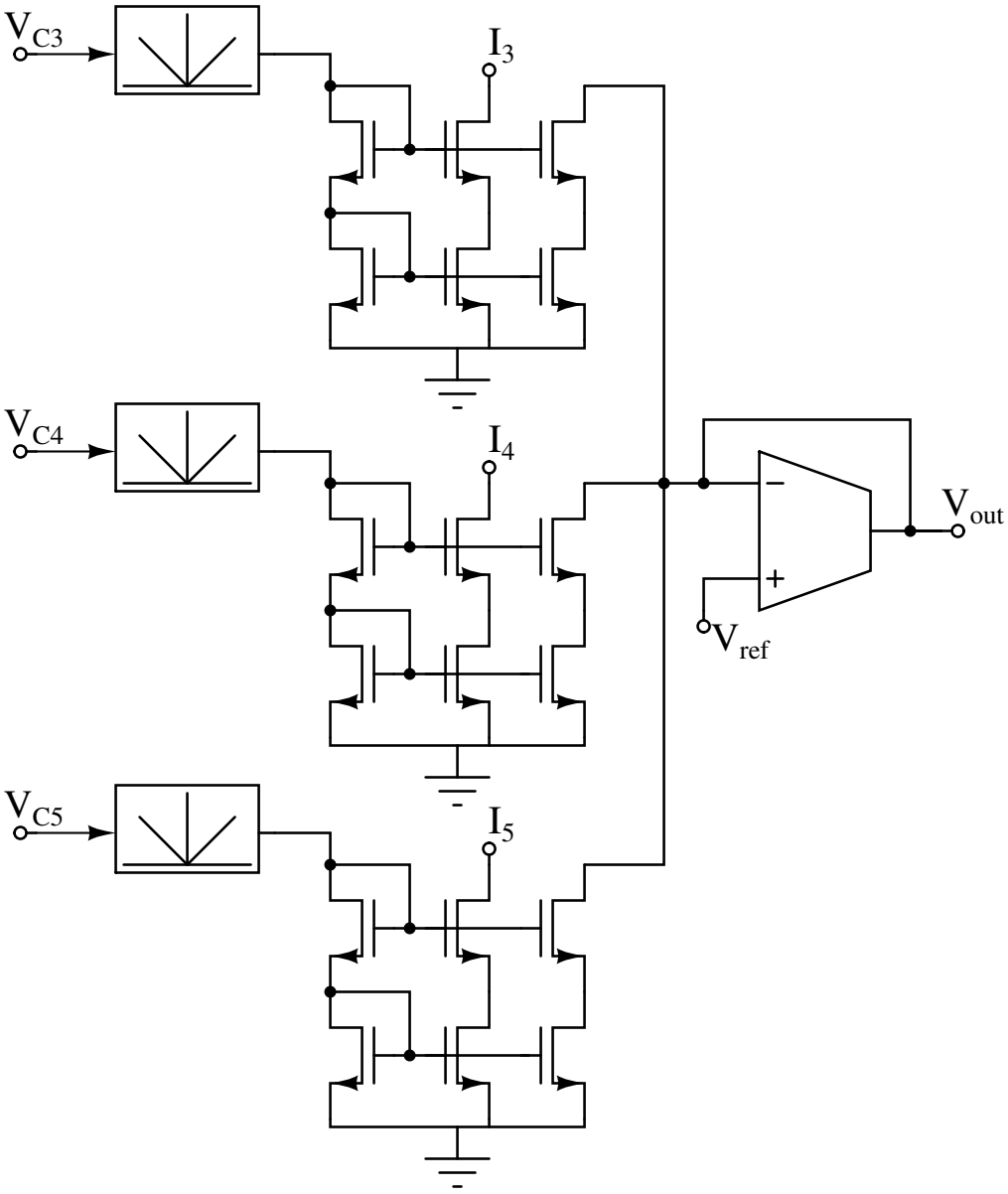


Figure 3.29: Circuit schematic of the whole computing unit.

that can be cut using instead a very low power current source. Besides, it would be better if the current of the *SBCS* (I_{REF}) is immune to variations in temperature and supply voltage (V_{dd}). The power consumption of the current source itself must also be minimized.

The chosen current source circuit schematic is shown in Fig. 3.33. This SBCS circuit was proposed by [6] and provides a good starting point for the given specifications. After having performed a theoretical analysis and many hand simulations, it was concluded in [15] that it was extremely complicated to find the optimum set of parameters for this current source, due to the great amount of possible combination of circuit parameters. Therefore, it was decided to use the automated approach described in this chapter.

For the optimization process, the length of transistors is used as parameters and the following three fitness functions were defined: reference current (I_{REF}), power consumption (P_{OUT})

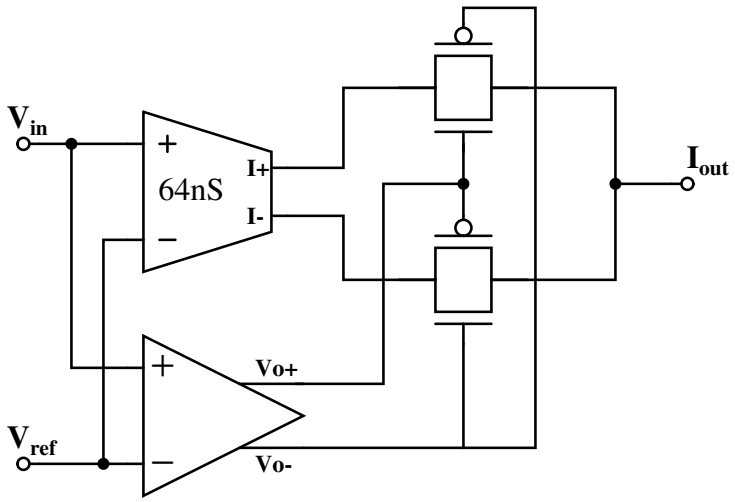


Figure 3.30: Circuit schematic of the current rectifier.

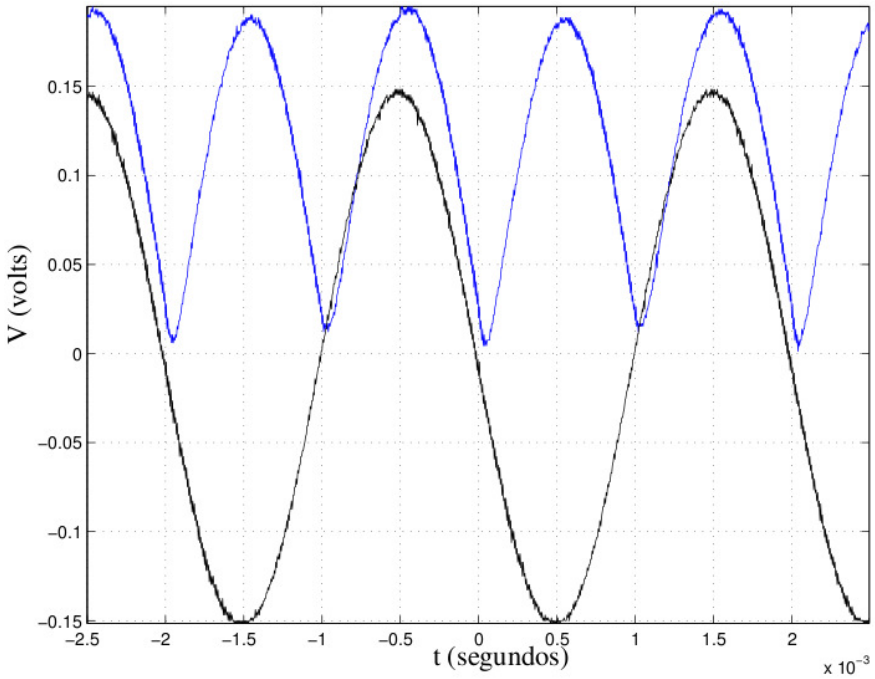


Figure 3.31: Rectifier unit output from hand-made version[9], for a sinusoidal input of 500 Hz, 150 mV peak. See the obvious asymmetrical response, product of both the systematic and random DC offset from the OTAs and the comparator. This asymmetry impacts the performance of the whole computing unit.

and the sensitivity of I_{REF} to supply voltage (V_{dd}). The sensitivity of I_{REF} to temperature as a fitness value as it is directly proportional to the dependency of I_{REF} to V_{dd}; so it is indirectly optimized due to the topology of the current source.

A stacking scheme of transistors has been used, namely *Stacking A* and *Stacking B*. Each stacking represents a different way of making transistor arrays, i. e. a series and parallel combination of transistors interconnected to form a bigger one (as proposed by[4]).

Two simulation environments were set up in order to assess the behavior of the circuit and

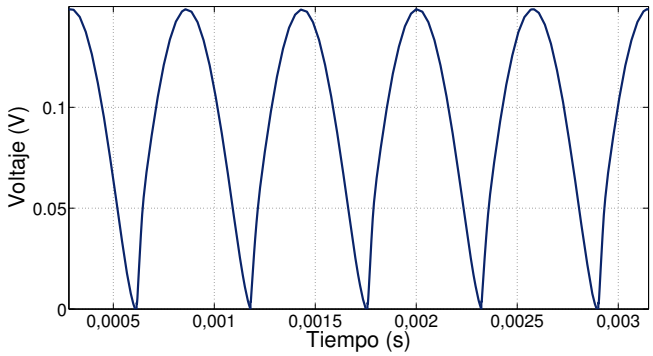


Figure 3.32: Rectifiers' output. A small distortion is still present near the tripping point of the rectifier. This distorition is, nonetheless, much smaller to that is seen in the original circuit (see [9]).

compute its fitness values. First a transient analysis to get the reference current and power consumption at the supply node and then a DC analysis to determine the reference current variation over a linear scan of supply voltage between 2 V and 4 V.

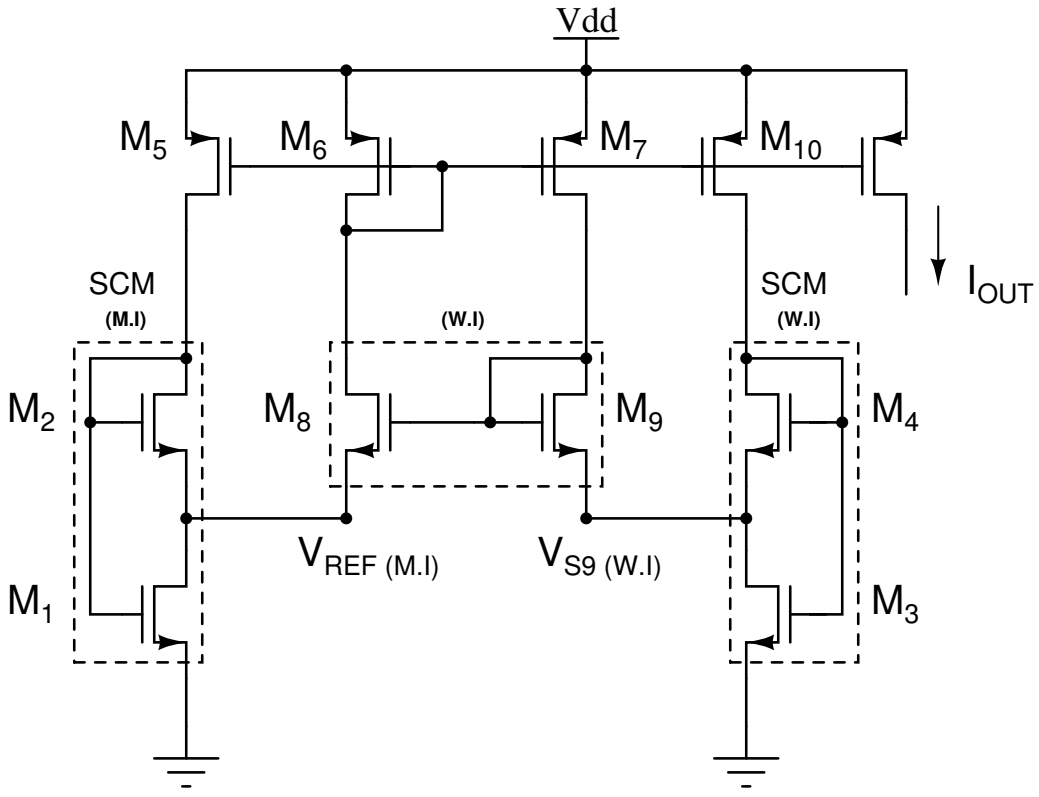


Figure 3.33: Circuit schematic of *SBCS* current source used for optimization, as proposed by[6].

Fig. 3.34 and Fig. E.1 show a graphic representation of the SBCS Pareto Front. It can be seen that the front corresponding to the scheme *Stacking B* has a greater amount of valid results. The optimization goal was that the reference current, output power and voltage sensitivity were as low as possible. However both figures show sharp changes on the surfaces, this may be an indication of potential unstable parameterizations and a possible source of errors. This

means that even a small change in a parameterization can cause a big change in the fitnesses of the circuit.

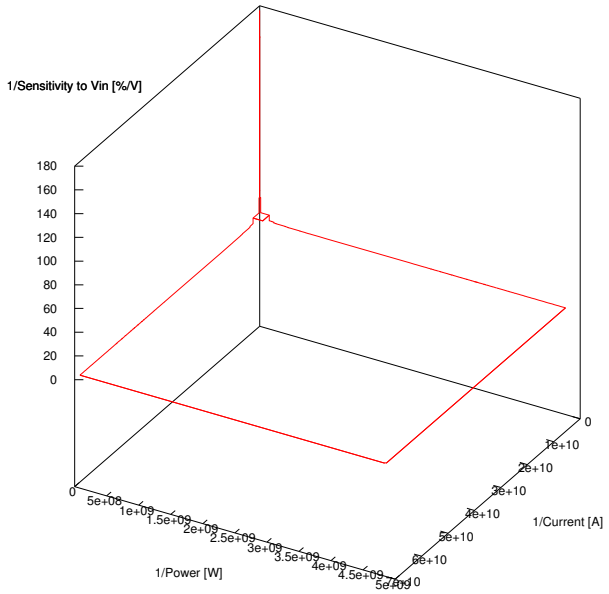


Figure 3.34: Pareto Front for optimization with *Stacking A*.

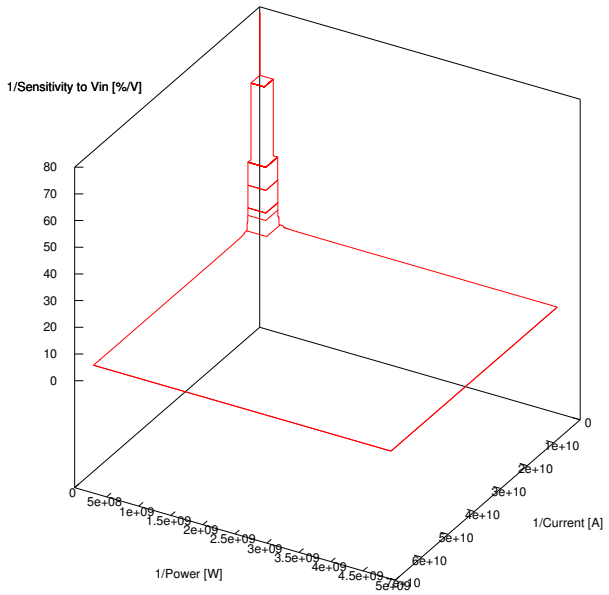


Figure 3.35: Pareto Front for optimization with *Stacking B*.

According to their fitness values, three prototype current sources were chosen for each stacking scheme. Table 3.18 summarizes the sources' main parameters. Sources *I* , *II* and *III*

correspond to *Stacking A* scheme, whereas sources *IV*, *V* and *VI* are related to *Stacking B* scheme.

Table 3.18: Best *SBCS's* obtained after optimization, based on schemes *Stacking A* and *Stacking B*.

Source	I	II	III	IV	V	VI
I_{REF} $[pA]$	245.64	303.95	337.99	254.68	217.33	218.46
P_{OUT} $[\eta W]$	4	4.96	5.52	4.13	3.11	3.54
Sens to V_{dd} $[\%/V]$	4.32	4	3.14	6.19	7.59	7.8
Sens to T $[\%/°C]$	0.3	0.26	0.49	0.37	0.77	0.81
Minimum area $[\mu m^2]$	28013.28	22890.04	48502.32	7902	22731.36	22776.4

Table 3.19 presents transistor dimensions for two selected *SBCS* current sources. It was selected one source per stacking scheme, based on their sensitivities and estimated areas.

Table 3.19: Transistor dimensions for two selected *SBCSs*.

	Stacking A		Stacking B	
Transistor	W $[\mu\text{m}]$	L $[\mu\text{m}]$	W $[\mu\text{m}]$	L $[\mu\text{m}]$
M_1	1.8	69.2	0.9	45
M_2	2.9	13.6	7	23
M_3	1.7	10.4	0.9	25
M_4	22.7	24.6	22	6
M_P	18.3	19.4	10	10
M_X	5.1	12.3	10	10

Inversion indexes for transistors in *SBCS* sources are shown in table 3.20 for both optimization processes. Both designs agreed with the theoretical analysis of [15] regarding the fact that M_1 should operate in moderate inversion. However, the rest of transistors operate in weak inversion, in disagreement with the same theoretical analysis. Optimization results show that transistor M_2 need not operate in moderate inversion in order to obtain a low-sensitivity constant reference current in the time domain, in contrast with what is asserted in [6]. Post-layout simulations gave positive results, except in the cases where the layout scheme proposed by [4] for the transistors was used, maybe because of excessive parasitics generated by the long metal lines needed to accommodate the stacks of transistors,a similar problem as the one already mentioned regarding the post-layout simulations of the rectifier's comparator in sub-section 3.2.3. As in the latter case, a more traditional multi-fingered approach to layout yielded results almost equal to those obtained with schematic simulation.

The fact that transistor M_2 operates in weak inversion and that no transistor operates in *triode* allows us to verify the power of genetic optimization in exploring the design space of a circuit. I also reveals the contrast between theory and experimental practice.

The resulting current source from the optimization of *Stacking B* scheme, has demonstrated to be the most robust in terms of the balance between its design parameters.

Table 3.20: *Drain* Currents (I_D) and inversion indexes (if) for *SBCS* design of *Stacking A* y *Stacking B*.

	Stacking A		**Stacking B**	
Transistor	I_D [pA]	i_f	I_D [pA]	i_f
M_1	450.39	14.69125	472.84	8.02390
M_2	242.71	0.96575	250.76	0.27963
M_X	477.90	0.09922	495.18	0.23342
M_3	243.49	0.00896	251.61	0.00116
M_4	245.64	0.02933	255.04	0.00433
M_P	238.40	0.01951	244.93	0.00260

Post-layout simulation results for the voltage variation of two of the sources are given in Fig. 3.36, to check for their sensitivity to V_{DD} variations. The final chosen prototype gives a reference current of only 254.7pA, scaled up to 2nA in order to supply it to the other units. The total comsumption is of 4.13 nW for a 4V V_{DD} supply, with a sensitivity to V_{DD} variations of 6.19%/V, and to temperature variations of 0.37%/°C, with a total area of 7902 μm^2.

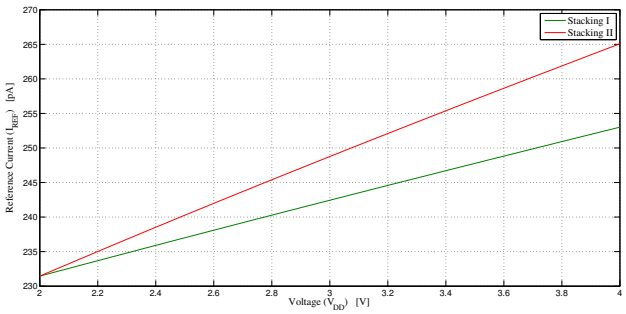


Figure 3.36: Current variation versus voltage variation for two of the three current source prototypes. Variation is slightly better for the first power supply (4.32%/V against 6.19%/V,), though its area is almost twice the second. In the end, the smaller source was chosen.

Chapter 4

Validation of the methodology through chip measurements

This chapter shows the process followed in the development of a test platform for the fabricated application specific integrated circuits (ASIC), which were optimized using the strategy described in this dissertation. As a second part of the chapter, measurement results are presented. The test and characterization of this ASIC will allow designers to determine if adjustments are required to meet design specifications and to verify the result of previous optimizations. The platform was designed to be scalable (both hardware and software) to allow users modify or expand the system as needed. It is possible to generate voltage or current signals for biasing, reference, or stimulus; and to set limits to ensure the integrity of the device under test (DUT). The software displays important information in real time and allows the user to save data to digital files and to apply data analysis. It was necessary to dimension and select the instruments to accomplish accuracy, precision, and scalability aspects. The software routines allow the user to configure the hardware, define test parameters and to visualize the test results. A custom printed circuit board was designed to avoid stray effects and to match with the shielding techniques of National Instrument's devices. Proper shielding represents the key aspect when measuring in the sub-nano amp range [12].

4.1 Measurement platform

This platform is based on the concept of virtual instrumentation (VI), where, by means of a computer, it is possible to acquire information from an instrument via a standard communications protocol (RS-232, GPIB, VXI, PXI). In older versions of these kind of systems, instruments were able to share information with a computer, but they were designed for a specific function only, in short, the instruments were not reconfigurable. Fig. 4.1 higlights the main differences between reconfigurable and non-reconfigurable instrumentation. It can be seen in the figure the flexibility that provides a reconfigurable system, providing modular hardware (instruments), shared resources and reconfigurable interfaces.

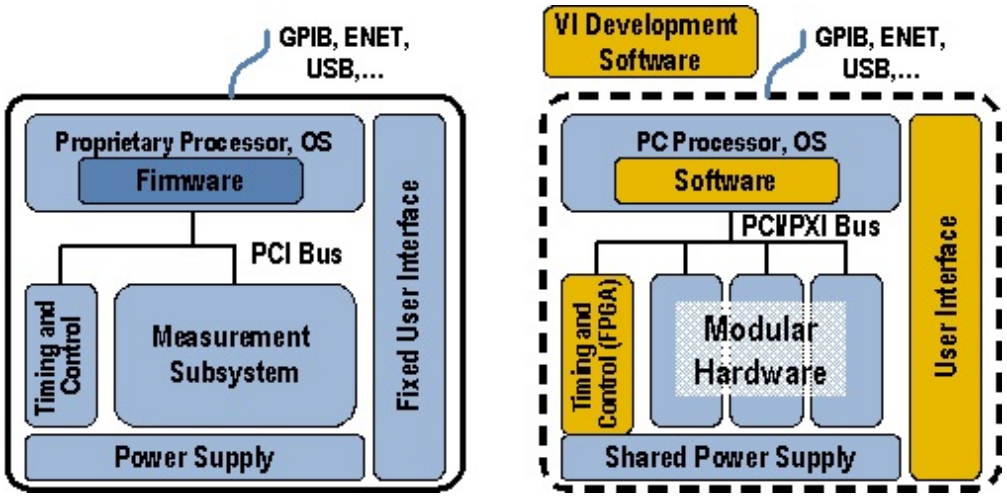


Figure 4.1: Traditional instrumentation (left) and software-based instrumentation,picture taken from [32]

In the following subsections, the three main components of the measuring platform are briefly described, namely: the source measurement units, the measuring board and the software routines.

4.1.1 Employed hardware

This deals with the selection of instruments required to perform the given task. Since the platform was provided by National Instruments, it was necessary to select from the wide variety of families and modules that this company has available. It was also necessary to take into account all the variables that needed to be measured and generated for the analog units (4 in total) that where included in the test chips. Features such as sampling rates, resolution, accuracy, shielding and signal conditioning were considered.

Fig. 4.2 shows a picture of the PXI platform. This platform was determined to comply with all of the aforementioned features. The requirements of the application are listed as follows: high processing capacity, resolution, accuracy and data transmission, as well as the capability of supporting virtual instrumentation, in order to facilitate the generation of test cases. The main objective of this process was to determine the number of signal sources, references, analog stimulus signals, maximum and minimum electrical magnitudes, number of analog outputs, such that the selected set of instruments would produce an accurate representation of the variables to be measured.

4.1.2 Printed circuit board (PCB)

The PCB was designed (by keeping continuity in connections and shielding techniques) to minimize the negative effects due to parasitic losses. Fig. 4.3 shows an example of the implementation of shielding in the printed circuit, this is mainly applied to the signals coming from the SMUs (Source Measurement Unit) as they were used to generate reference-

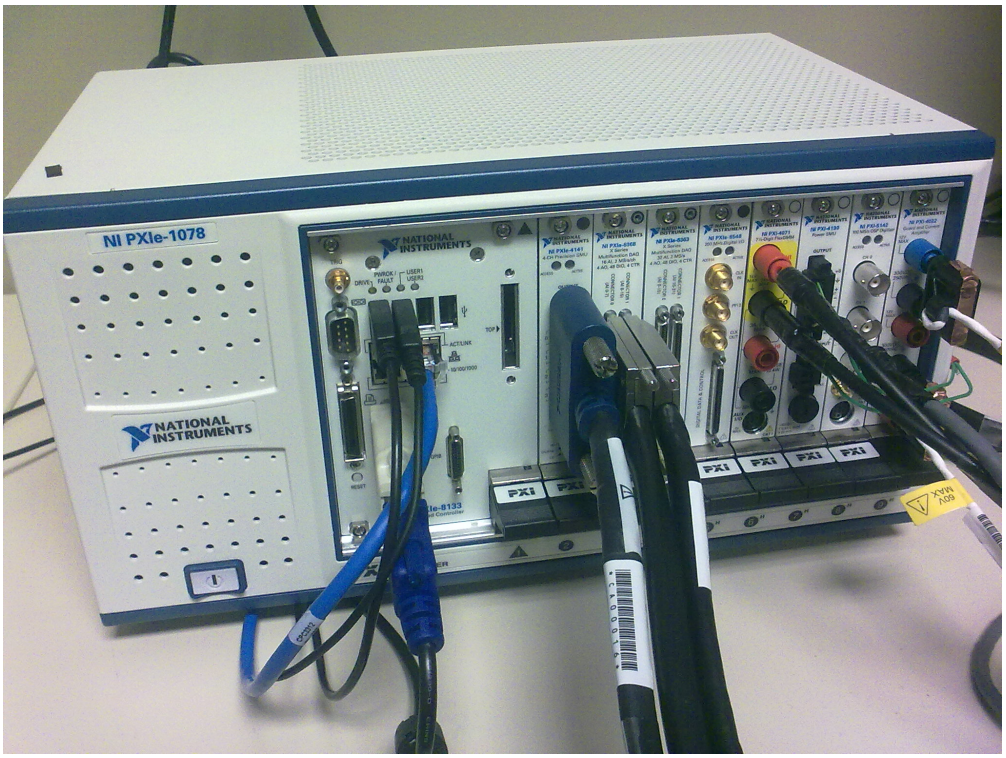


Figure 4.2: PXI platform

voltages and very-low-level currents to bias many circuits within the chip area. Therefore, the transmission lines that carry these signals had spread throughout the whole ASIC, becoming signals vulnerable to interference (noise) and attenuation.

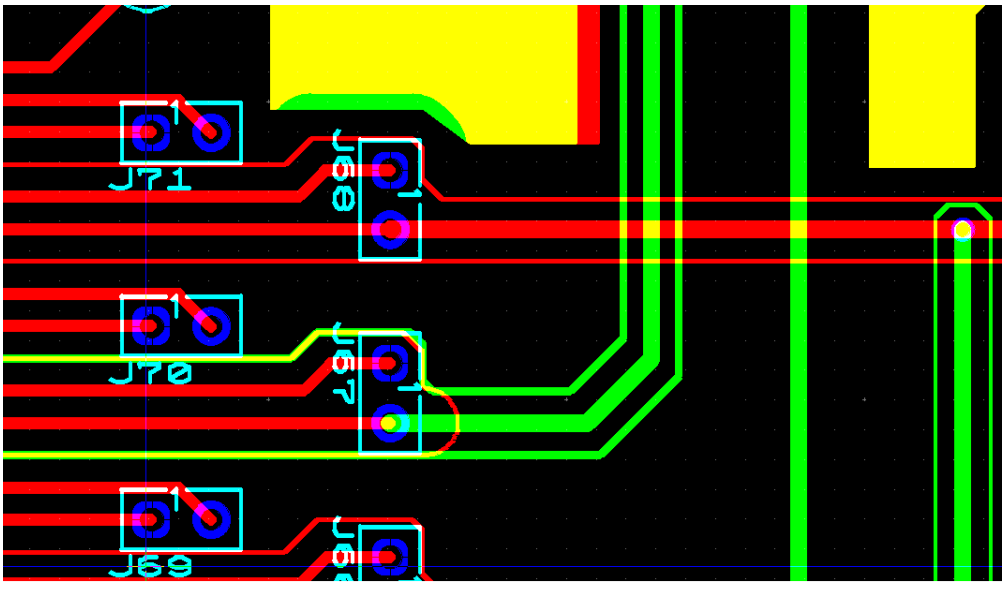


Figure 4.3: Shielding techique implemented in the PCB (red: tracks on top layer, green: tracks at bottom layer, yellow: layer overlap), also thin green and red tracks are guardlines connected to the SMUs.

The NI PXIe-4141 module has dedicated terminals for guard lines, so shield lines on the

printed circuit were connected to these terminals in order to give continuity to the shielding technique used in both the module and the cable (tri-axial cable). However, due to the amount of connections required for the interconnection circuits , test leads and selector switches, sometimes there was overlapping between lines carrying different signals. In these cases, the crossings between lines were performed at 90° angles . This technique is used in copper transmission lines (usually radio frequency circuits) because the overlap area is minimized, which in turn reduces electromagnetic induction and the capacitance of the crossing point. It must also be pointed out that the operating frequency of the signals measured is less than 1000 Hz in all cases, therefore avoiding possible interference from guard lines.

Fig. 4.4 shows the PCB photograph, already embedded in casing.

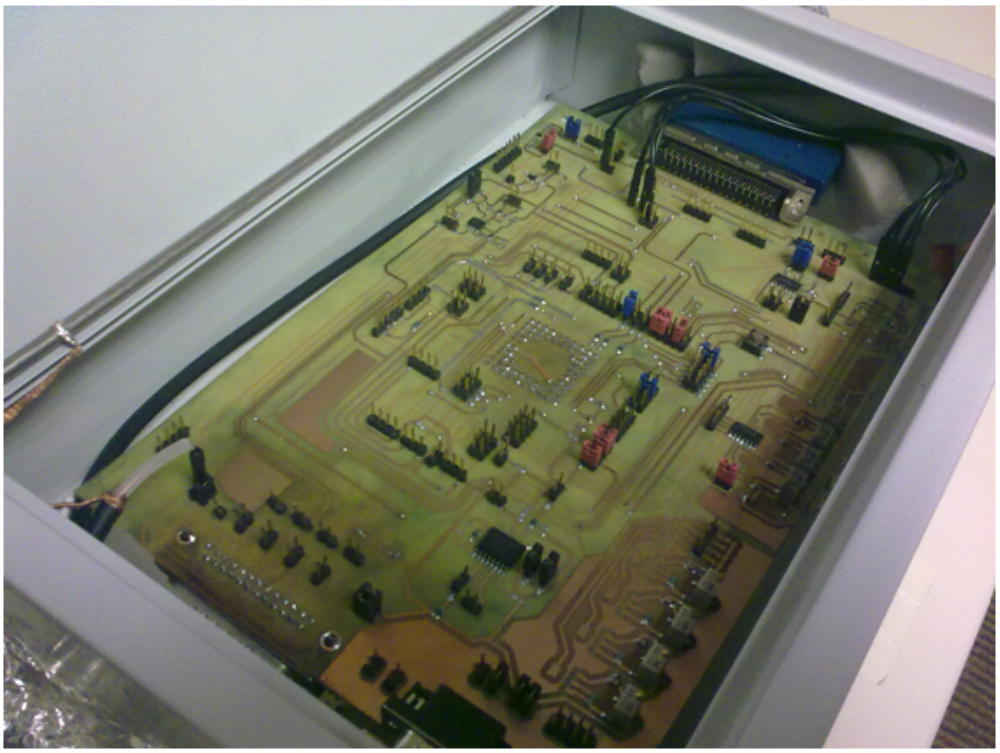


Figure 4.4: PCB for testing, a two-layer board was used.

4.1.3 Software

The LabVIEW graphical programming environment is used in order to generate hardware configurations, circuit signal measurement setups, display of results and data storage. The process of programming in LabVIEW is not very different from traditional imperative programing languages, in the sense that input and output variables must be defined, there are data typesand the algorithm to be programmed has to be properly structured. A main routine was generated for every circuit to be tested.

4.2 Measurements perfomed on fabricated circuits

4.2.1 OTA experimental results

In order to test the operation of the OTA, a voltage sweep was carried out at the differential input of the circuit, and the value of the output current was measured, for each step. This allows to see the deformation of the signal due to the the OTA's nonlinearities, i.e. when the amplitude of the input reaches the limits of the linear range. The transconductance curve is obtained by automatically taking the first derivative of the output current curve, a task that is facilitated by the virtual instrumentation environment used.

Figs. 4.5 and 4.6 show measurements of the transconductance vs input voltage and the output current as a function of input voltage, respectively. Both are satisfactory results that allowed us to verify the effectiveness of our optimization methodology. For instance, the transconductance (taken as 5% over the value obtained when the input is equal to the reference voltage of 2V) has a value of 84.38nS. This represents an error of -0.73% from the value proposed in [31]. Nevertheless, a 20 mV offset can be seen in Fig.4.6, which may be explained by the missmatch of transistors at the input differential pair of the amplifier. The optimization methodology did not consider missmatching effects at this point, later this subject will be addressed. From the same figure, the linear range is aproximately equal to 1V.

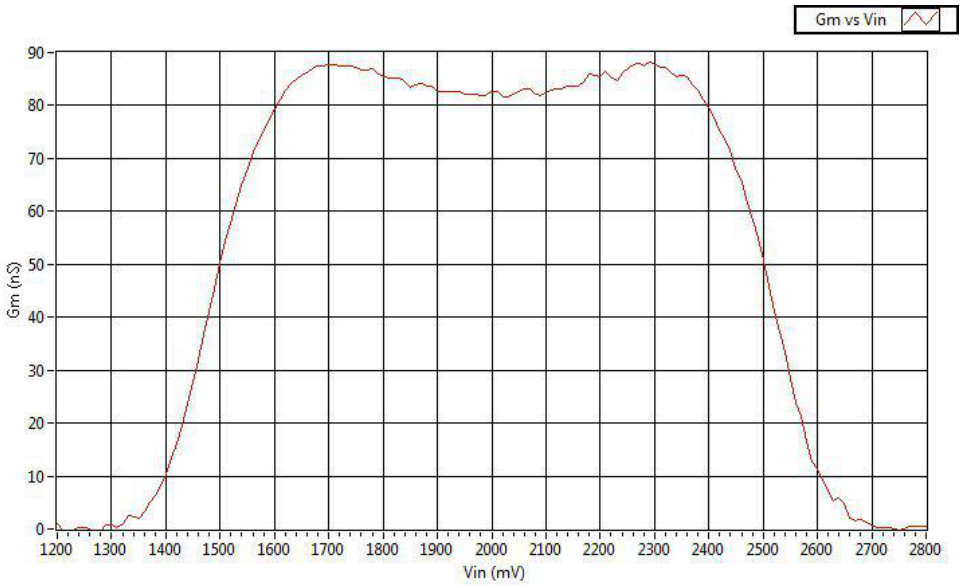


Figure 4.5: Measured G_m curve as a function of the input voltage

4.2.2 SBCS experimental results

Fig.4.7 shows the outputs of the two dual self-bias current sources fabricated. Every dual output had to be symetric, however it is seen an offset for each output of the source. The dif-

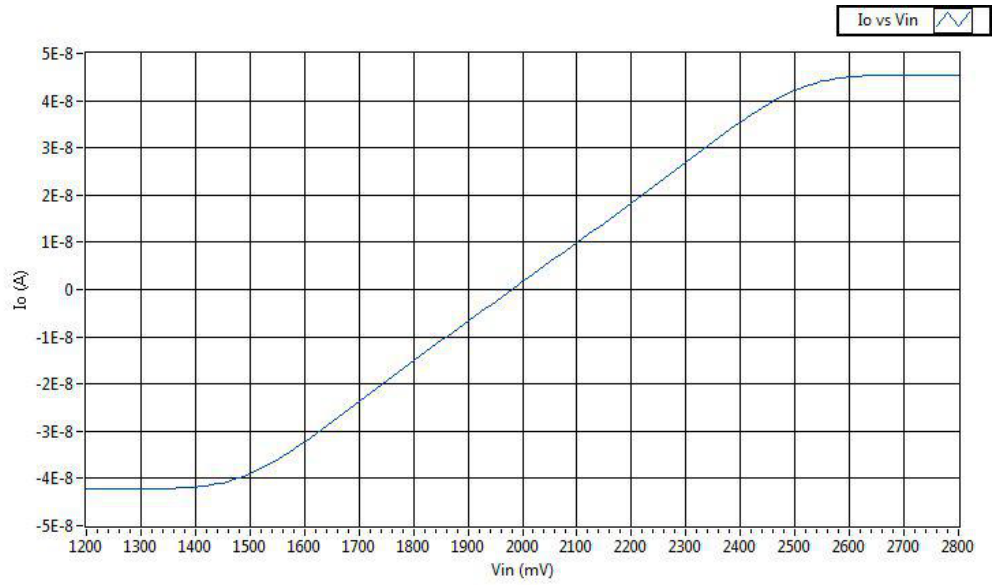


Figure 4.6: Measured output current as a function of the input voltage

ference in reference currents, as shown in Fig.4.8, can explain this offset between the outputs of the SBCS, as the output currents are scaled-up and copied versions of the reference.

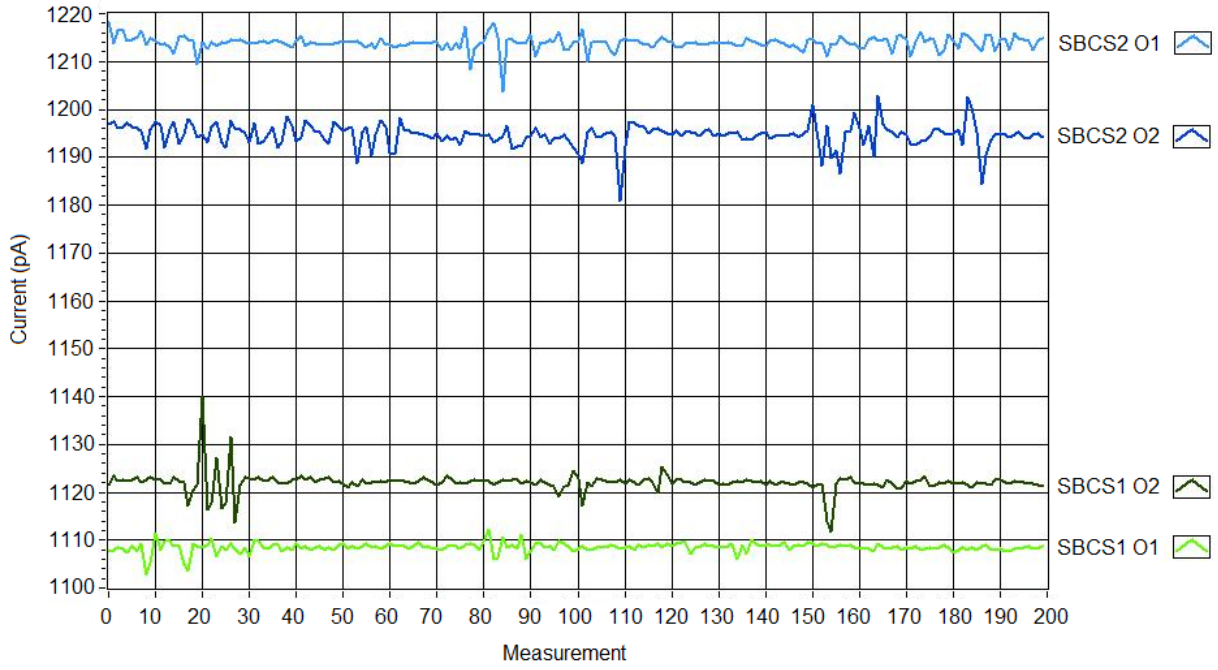


Figure 4.7: Dual SBCS output currents

Table 4.1 summarizes the output current values, it was expected to get 2 nA currents.

Table 4.2 shows high error percentages in both sources. A possible reason for this behaviour can be obtained by looking at the Pareto Fronts for these two circuits. Figs. 3.34 and E.1 show drastic changes between the points in the Front, this means a little change in the

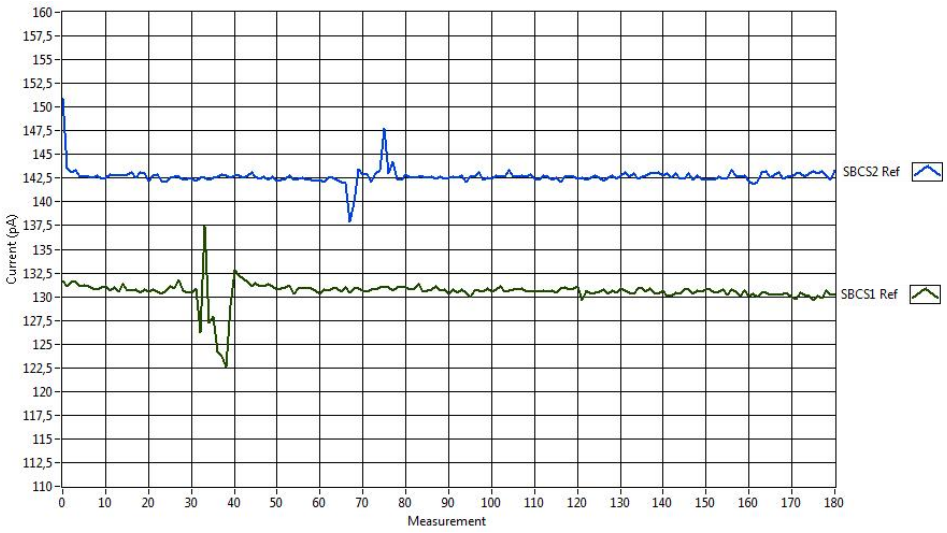


Figure 4.8: Reference currents in SBCS

Table 4.1: Output values for each *SBCS*.

Source	I Ref (pA $\pm 10pA$)	Iout_1 (pA $\pm 10pA$)	Iout_2 (pA $\pm 10pA$)
SBCS1	130,5	1108,5	1122,0
SBCS2	142,4	1214,0	1195,0

parameters (W and L for each transistor) can result in a considerable difference in the fitness values. In these cases it is recommended to use a different current source architecture such that this situation can be attenuated. Both current sources also presented start up problems, in fact, it was needed to bias and restart the system repeatedly in order to make it work. I was not observed any start up time pattern.

Table 4.2: Error percent for each *SBCS*.

Error	I Ref (%)	Iout_1 (%)	Iout_2 (%)
SBCS1	-48,41	-44,80	-43,90
SBCS2	-43,71	-39,31	-40,27

4.2.3 GmC filter

Fig. 4.5 shows the time response for coefficients 3, 4 and 5. This response was obtained by applying a 600 Hz sinusoidal stimulus of 50mV magnitude with a 2.0V offset. This test allows to check for proper filter operation and see any offset present at the ouputs, which are shown in table 4.3.

Fig.4.10 shows the frequency response of the GmC filter. In Fig.4.11 it can be seen that the offset is constant for the whole frequency range of the test. In order to define the range of frequencies, a 10% of Total Harmonic Distortion (THD) was used as reference.

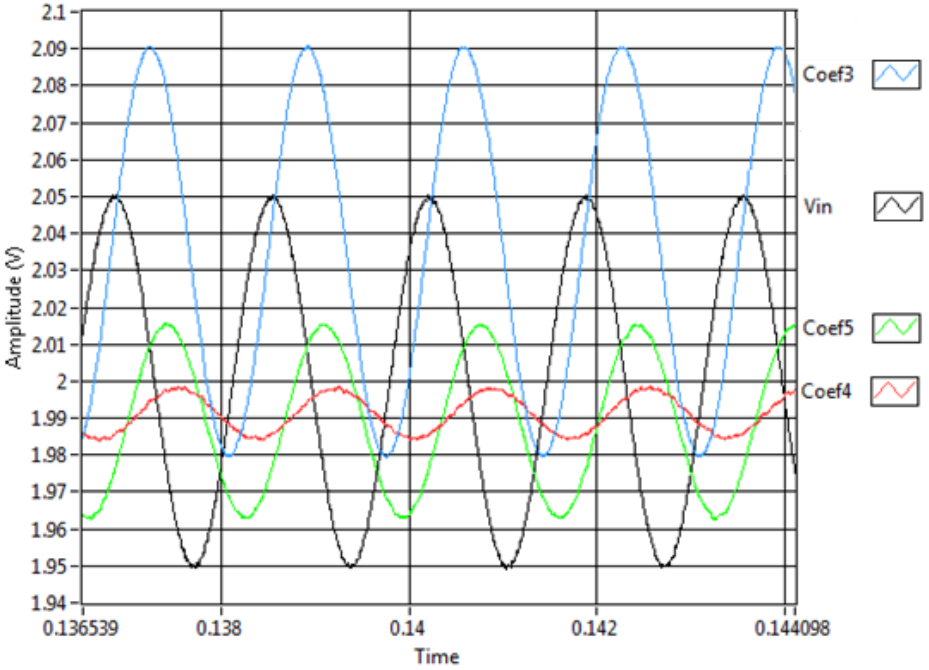


Figure 4.9: Filter time response

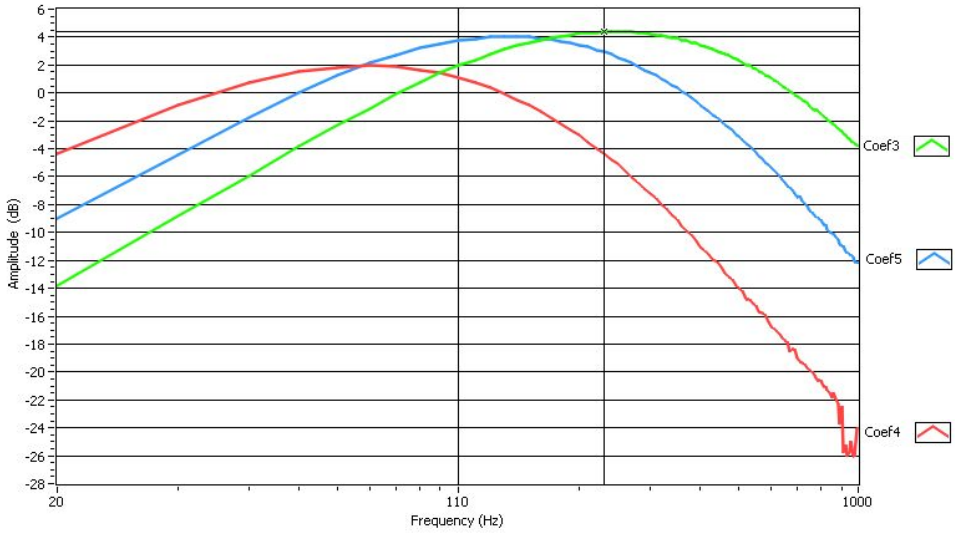


Figure 4.10: Filter frequency response

Table 4.3: Output voltages and offsets for the GmC filter

Signal	Amplitude (mV)	Offset (mV)
Coeff3	56,14	34,63
Coeff4	7,74	-8,72
Coeff5	26,88	-11,09

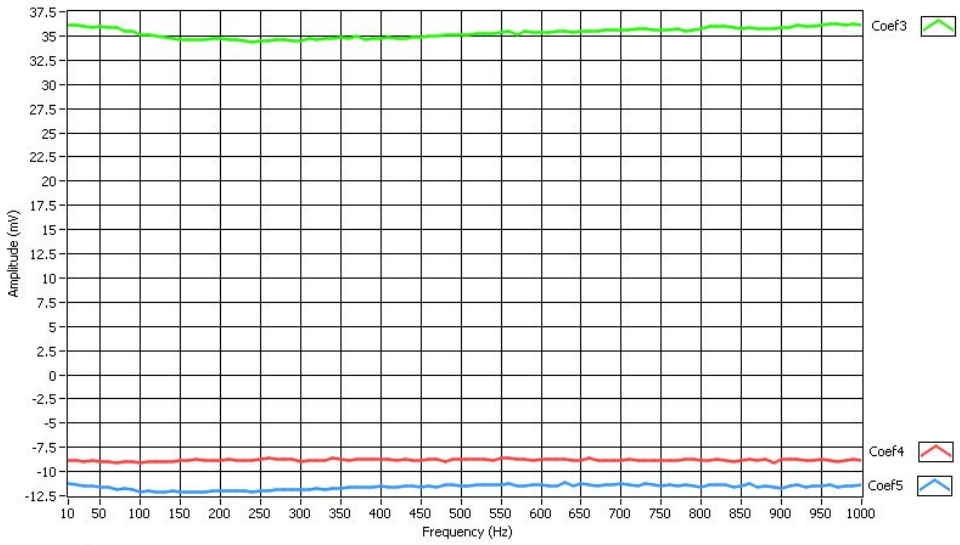


Figure 4.11: Filter coefficients offset

4.2.4 Computing unit

The response of the computing unit is shown in Fig.4.12. A sweep voltage was applied to each of the inputs representing coefficients 3, 4 and 5. The left hand side of the figure shows the output when activating only one of the inputs, keeping the two others fixed to the offset voltage (2V). The right hand side graph shows the output when applying the same signal to the three inputs. The linear range for both outputs is 980mV and 750mV, respectively.

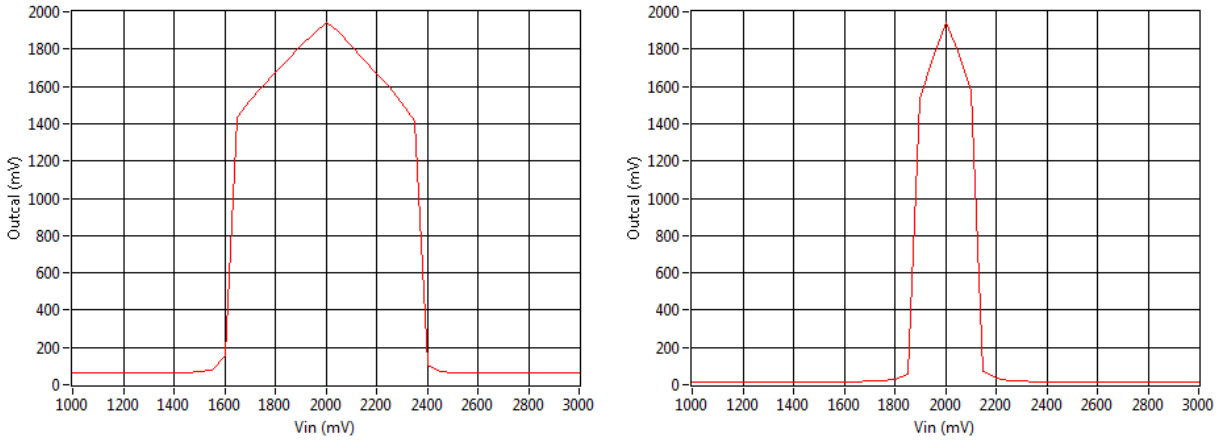


Figure 4.12: Computing unit response

Chapter 5

Circuit optimization considering mismatch

Transistor mismatch, understood as the difference in performance of two or more devices identically layed out in the same integrated circuit, is key to precision analog design. In this chapter, it is defined the strategy to be applied for the analysis of variability in selected designed structures. It follows the model proposed in [17] and [46] as it characterizes mismatch in the same domain (currents and voltages) and using the same tools and models common in the design of a circuit, ie. standard Spice-like simulators using BSIM models. Although approaches like Monte Carlo have enjoyed widespread usage for statistical analysis of circuits affected by technological variations [53], it requires a large number of simulations for every point in the parameter space. This would be impractical to implement within the methodology presented in this dissertation, due to its excessive consumption of cpu-time.

5.1 Mismatch modeling

In order to include the effect of process variability into the optimization flow, the Pelgrom model [46] for the technology in question (0.5μm) was used to extract the parameters of mismatching. According to this model, there are mainly two parameters that contribute to mismatch, one is the difference between the threshold voltages (δV_t) of a pair of MOS transistors, and the other is the variability of the current factor ($(\delta\beta/\beta)$). The first is due to the uncertainty in the number of doping atoms in the depletion layer, the second is due to variability in the mobility of charge carriers. We relie in the statistical analysis that is performed by the process foundry to approximately quantify mismatching effects, thus reducing design errors. Figs. 5.1 and 5.2 show the corresponding statistical graphs which can be used to determine the standard deviation of these parameters. The real data coming from the technology vendor cannot be disclosed, therefore these figures are taken from [46] and [45].

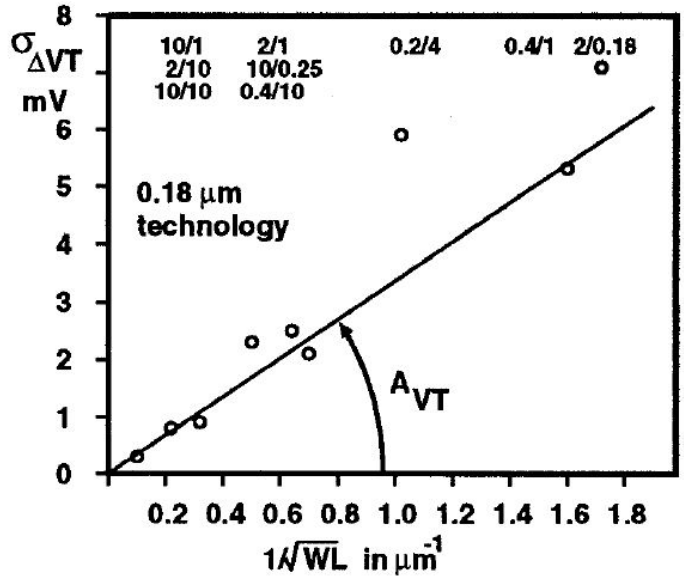


Figure 5.1: Standard deviation of V_t versus the square root of the inverse area [46]

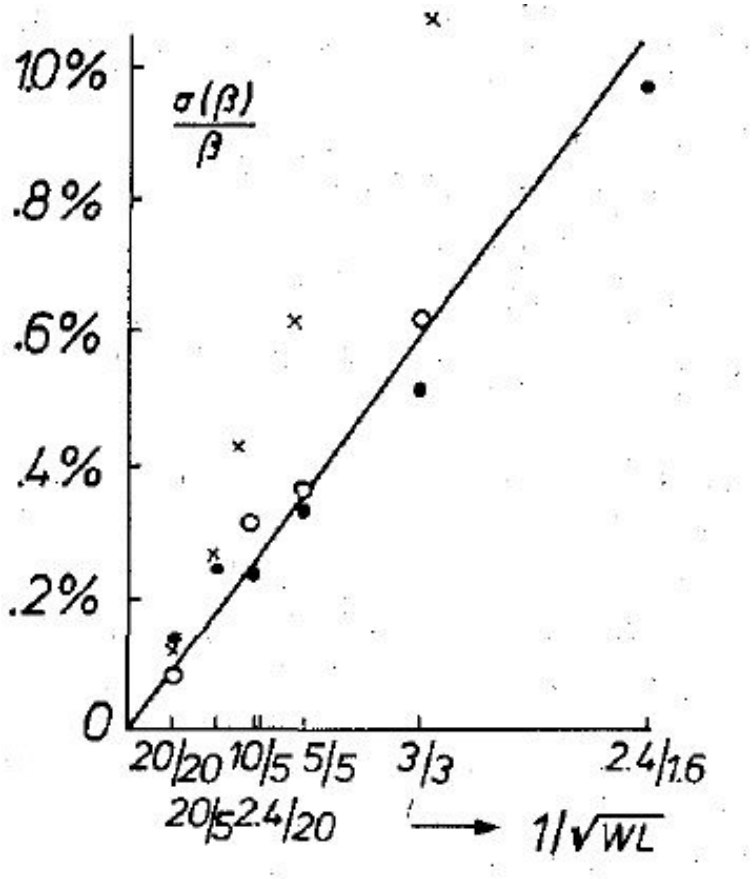


Figure 5.2: Percent Standard deviation $\delta\beta/\beta$ versus the square root of the inverse area [45]

In order to determine the effect of mismatching from these graphs, the classic Pelgrom model equations are used. The expression

$$\sigma(\delta V_t) = \frac{A_{VT}}{\sqrt{WL}} \tag{5.1}$$

indicates how to calculate the standard deviation of the threshold voltage variability ($\sigma(\delta V_t)$).

The percentage of mismatching due to the variability in the current factor ($\delta\beta/\beta$) is described by

$$\%Mismatching_{\delta\beta/\beta} = \frac{A_{\delta\beta/\beta}}{\sqrt{WL}} \tag{5.2}$$

Since the optimization algorithm generates, among other things, widths and lengths of transistors, equations 5.1 and 5.2 become very useful to derive figures of merit regarding mismatch.

5.2 Mismatch aware design process

An Operational Transconductance Amplifier was used as an example of how to apply the optimization methodology considering process variability. The architecture of this OTA is shown in Fig.5.3. This stage of the methodology consisted in preparing the optimization tool to appropriately size the OTA, according to the limitations imposed by the Pelgrom model and respecting the requirements of the specific application at the same time. Therefore, C/C++ routines were added in order to compute the new fitness functions, which are described in the following:

1. Average power consumption, this value is to be minimized. It is computed using the general expression:

   $$P_{Avg} = IV \tag{5.3}$$

   Figure 5.4 presents the flow diagram which was implemented to compute the average power through the node V_{dd}.

2. Transconductance: it was computed following its definition:

   $$G_m = \frac{\partial I_{out}}{\partial V_{in}} \tag{5.4}$$

   Figure 5.5 describes the corresponding computation process, where a curve $I_{out}vsV_{in}$ is generated and taken its first derivative, resulting in the curve G_mvsV_{in}.

3. Systematic offset. For this calculation, it was used the same curve $I_{out}vsV_{in}$ in order to determine the current value where the input voltage is equal to the voltage reference. From this data, the systematic offset is calculated with the expression

   $$V_{Offset} = \frac{I_{out} \mid Vin = Vref}{G_m} \tag{5.5}$$

   Figure 5.6 describes the corresponding computation process.

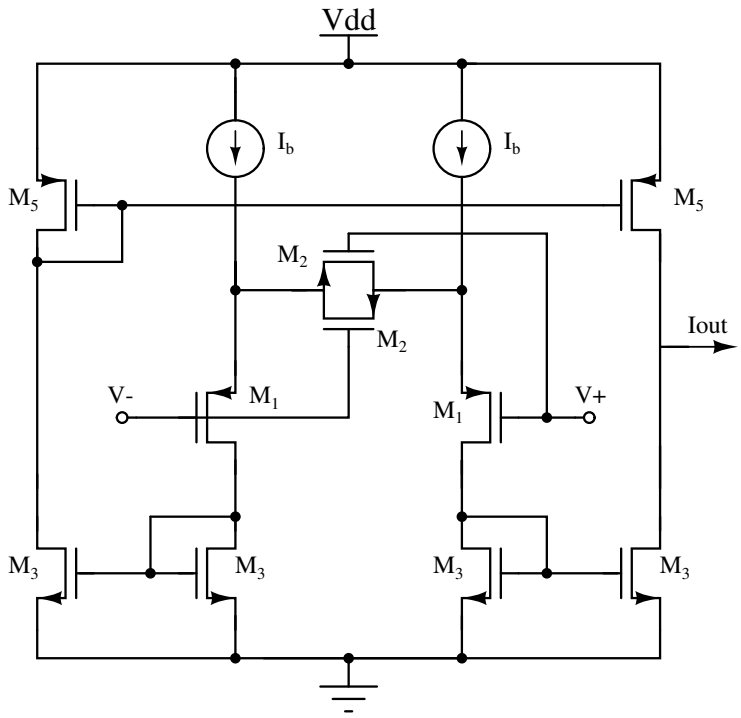


Figure 5.3: Circuit schematic for the OTA used for mismatch analysis.

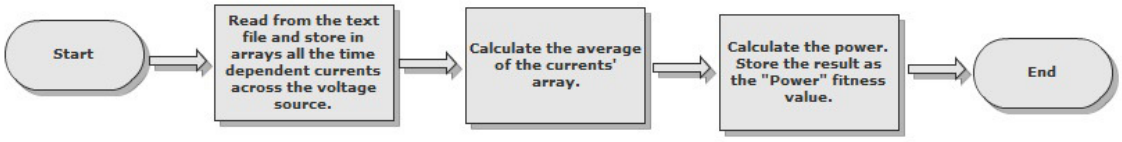


Figure 5.4: Average power computation flow diagram

4. Linear range: this value is to be maximized. To determine it, the graph $G_m vs V_{in}$ was used, in which the linear range is the maximum difference between the two input voltages whose Gm is not less than 5% its maximum value. Figure 5.7 describes the corresponding computation process.

5. Bandwidth(BW). The filter in which the OTA is intended to be used must operate in the range of $70Hz - 200Hz$, so the recommendation is that the OTA's BW should be at least a decade bigger than the filter's. For this reason, the larger the BW, the better. It is calculated setting up an AC analysis, and reading the BW from the simulator's

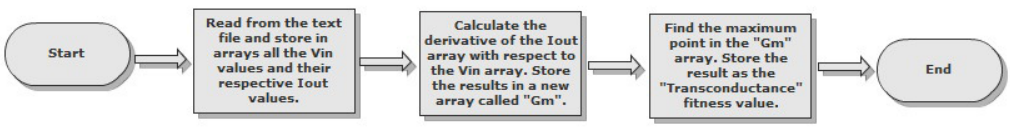


Figure 5.5: Transconductance computation procedure

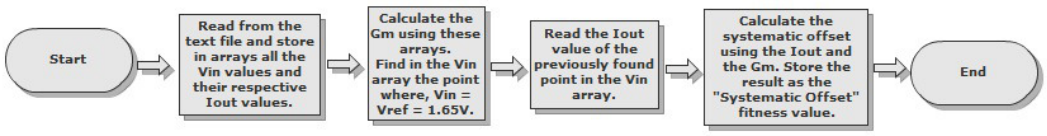


Figure 5.6: Offset computation procedure

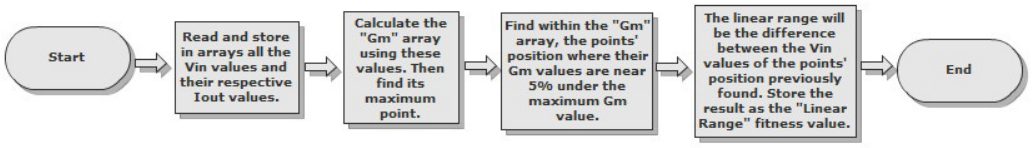


Figure 5.7: Linear range computation procedure

text output. Figure 5.8 explains this process.

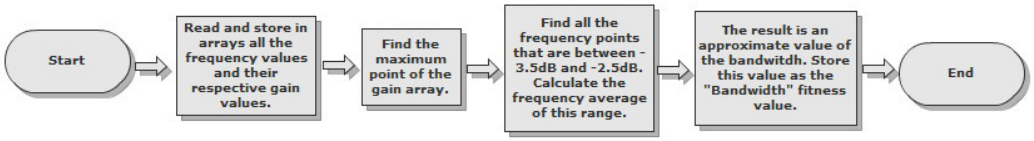


Figure 5.8: Bandwidth computation procedure

6. Slew Rate (SR) To calculate the SR, a capacitor was placed at the output as a load during a transient simulation and the time taken to charge and discharge this capacitor was recorded. Then the following expression is applied:

$$SR = \frac{dV_{out}}{dt} \mid max \tag{5.6}$$

   Figure 5.9 describes the corresponding computation process.

7. Input capacitance. This parameter is obtained by reading the output report file of the simulation and searching for tha total capacitance seen at a specified node. Figure 5.10 describes the corresponding computation process.

8. The standard deviation of the threshold voltage variation, (δV_t). This fitness value was added in order to reduce the offset in the OTA due to mismatching in the differential transistor pair. The calculation was performed reading the dimensions of the transistor and then applying the expression 5.1, with Pelgrom model's cofactors provided by the manufacturer.
   Figure 5.11 describes the corresponding computation process.

9. Variability of the current factor $((\delta\beta/\beta))$. This fitness was added to mitigate the offset produced by mismatching, by choosing good parameterizations according to this

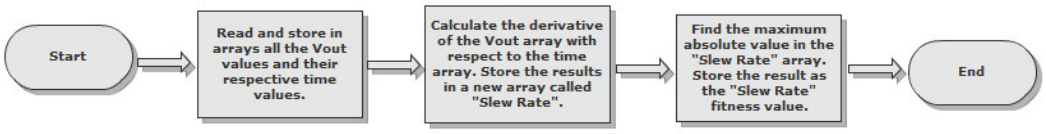


Figure 5.9: Slew rate computation procedure

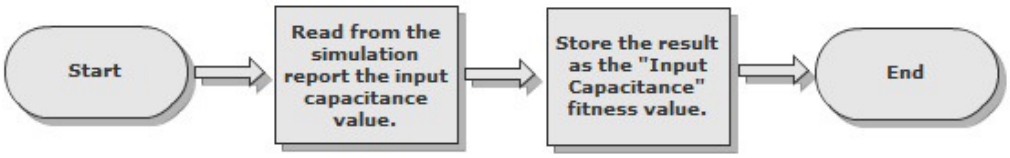


Figure 5.10: Input capacitance computation procedure

fitness. This parameter was calculated for transistors M_3 and M_5 because current mirrors are much more negatively affected by $\delta\beta$. In the same way as in the previous fitness value, this one was calculated reading the sizes of transistors and using the equation 5.2 along with Pelgrom model's cofactors provided by the technology vendor. Figure 5.12 describes the corresponding computation process.

5.3 OTA optimization considering mismatch

Table 5.1 shows the initial, and the minimum and maximum values for each parameter of the OTA.

For this case, the optimization tool was set up and run to generate a 5000-point Pareto front. Since this front has more than three dimensions, it is not possible to generate a single graph. Therefore a program was written to be able to extract from those 5000 points the parameterizations that best fit the design specifications.

Table 5.2 shows the corresponding unitary transistor dimensions, with a bias current of 20 nA.

With the dimensions of the table 5.2, functional tests for the OTA were performed. The

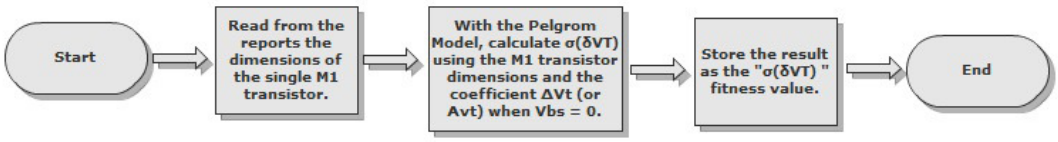


Figure 5.11: Standard deviation of the threshold voltage variation, (δV_t) computation procedure

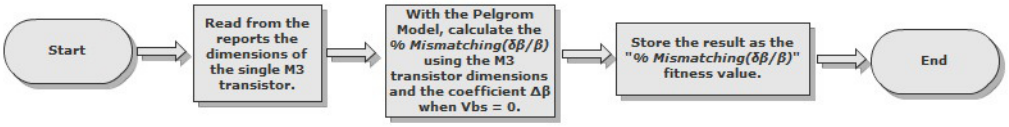


Figure 5.12: Current factor $((\delta\beta/\beta))$ computation procedure

Table 5.1: Parameters of the Amplifier.

Parameter	Initial values	Variation span
$L_1(\mu m)$	1.6	1.2-15
$W_1(\mu m)$	9.6	1.2-15
$L_3(\mu m)$	5	1.2-15
$W_3(\mu m)$	9.1	1.2-15
$L_5(\mu m)$	9.6	1.2-15
$W_5(\mu m)$	5	1.2-15
I_b(nA)	20	15-20

Table 5.2: Unitary transistor dimensions for the OTA, considering variability.

Transistor	W(μm)	L(μm)
M_1	1.6	10
M_2	9.6	10
M_3	5	4
M_5	9.1	9.1

negative input (V-) of the OTA was set to the reference voltage, whereas in the positive input (V+) a configurable source was connected in order to perform different types of anaysis: DC, AC or transient. The following figures show the corresponding electrical characteristics. Figs. 5.13 and 5.14, represent the results of performing a sweep in the source DC input variable. The first graph, correspons to the circuit's characteristic curve, while the second one corresponds to the derivative of the curve Iout vs. Vin, which could determine the transconductance, as already explained in 5.2.

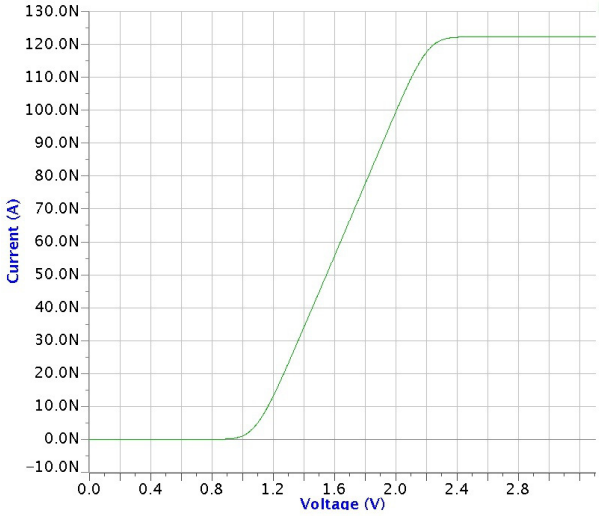


Figure 5.13: OTA's output current as a function of input voltage

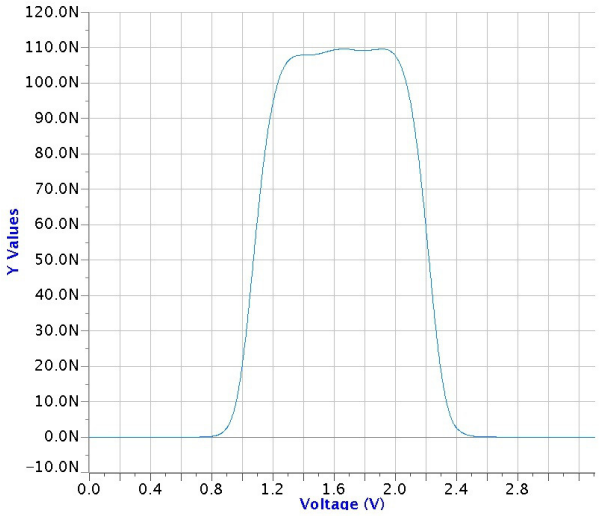


Figure 5.14: Transconductance as a function of input voltage

An AC analysis was carried out in order to check the amplifier's frequency response, which

is shown in Fig. 5.15. The magnitude response is green and the yellow trace is the phase.

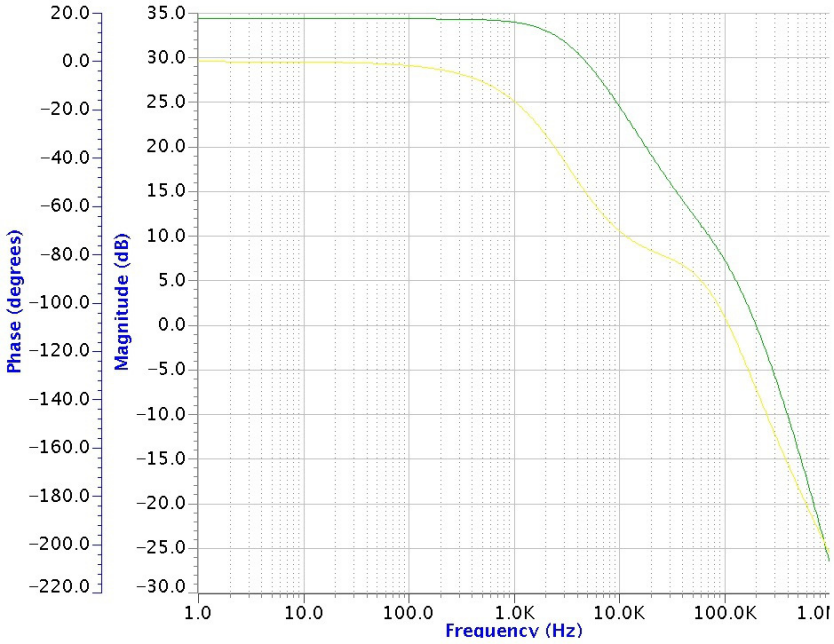


Figure 5.15: OTA's frequency response

Finally, the transient response of the OTA was obtained, the variable source corresponds to a sinusoidal signal with a magnitude of 20 mV at 1kHz, with a 1.65V reference and a capacitor placed as a load at the output. Figs. 5.16 and 5.17 show the results.

As a summary of the previous results, Table 5.3 shows the performance features of the selected OTA parameterization.

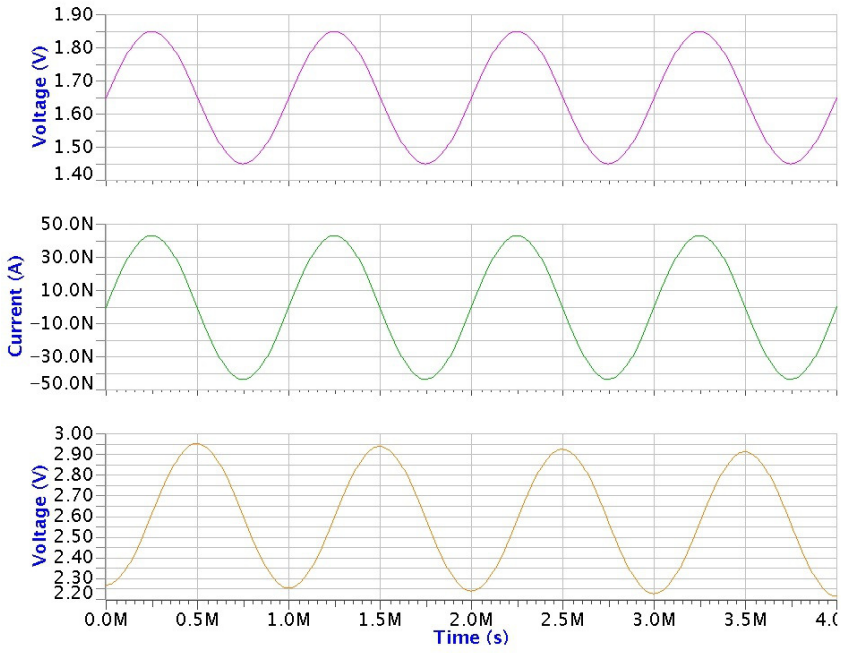


Figure 5.16: OTA's transient response. Top down:input voltage, output current and output voltage.

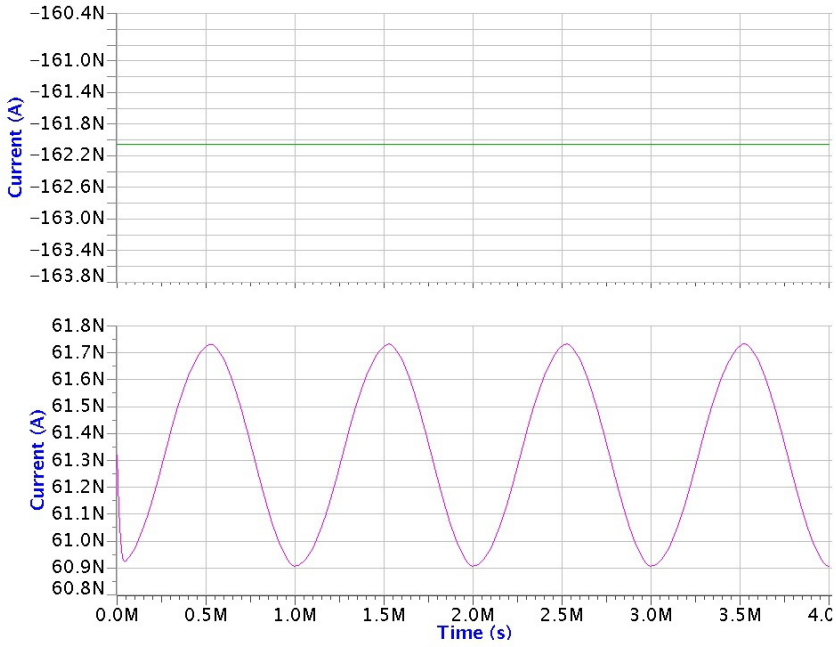


Figure 5.17: Current consumption(top) and offset current (bottom) for the transient response

Table 5.3: Performance features for the OTA, considering variability.

Performance feature	Value
Power consumption	535 nW
G_m	110 nS
Linear range	0.86 V
Sistematic offset	0.55 V
Slew rate	1794 V/s
Bandwidth	2.8 kHz
C_{in}	746 fF
$\sigma(\delta V_t)$ M_1 and M_2	4.725 mV
$\%Mismatching_{\delta\beta/\beta}M_3$	0.21%
$\%Mismatching_{\delta\beta/\beta}M_5$	0.96%

Chapter 6

Conclusion

This dissertation has introduced a methodology that uses a heuristic method as an effective way of designing and optimizing integrated circuits. The proposed methodology is automated which means that it avoids the designer to spend valuable time adjusting a circuit to meet the specifications. Moreover, the task of optimization has been treated as a multi-objective problem, where the Pareto front becomes the tool of analysis for trade offs between fitness functions . In fact, the nature the problem in question (multiple conflicting goals) imposes the multi-objective approach. Specifically, it proves to be an effective methodological approach for adjustment and improvement of analog circuits especially used in fire detection systems in the tropical forest. The approach presented here not only greatly reduced the time required to design and simulate such circuits, but also allows for optimal solutions or, strictly speaking, a good approximation of the optima. The design based on manual calculations lacks the advantages just mentioned. It was also shown how the methodology is suitable for design and optimization of digital circuits, namely Current Mode Logic was taken as a demontrative example. A test chip with several analog building blocks was fabricated in order to validate the optimization flow.

Future lines of research include making the layout level be part of the optimization flow, as automated layout of analog cells is very limited in the EDA industry. Another focus of research is addressing more profoundly the effect of mismatch on the fitnesses of a circuit, here the problem was partially solved by computing fitness functions derived from mismatch parameters. However, the subject remains open.

Bibliography

[1] Pablo Alvarado. *Segmentation of color images for interactive 3D object retrieval*. PhD thesis, Fakultät für Elektrotechnik und Informationstechnik der Rheinisch-Westfälischen Technischen Hochschule Aachen, 2004.

[2] Pablo Alvarado, Néstor Hernández, and Marvin Hernández. D2ARS: Diseño y desarrollo de aplicaciones basadas en redes de sensores. Propuesta de Proyecto, Vicerrectoría de Investigación, ITCR, Octubre 2007.

[3] A. Arnaud. *Very Large Time Constant Gm-C Filters*. PhD thesis, Instituto de Ingenieria Electrica - Facultad de Ingenieria, Universidad de la Republica, Montevideo, Uruguay, April 2004.

[4] A. Arnaud, R. Fiorelli, and C. Galup-Montoro. Nanowatt, sub-ns otas, with sub-10-mv input offset, using series-parallel current mirrors. *Solid-State Circuits, IEEE Journal of*, 41(9):2009 –2018, sept. 2006.

[5] H. I. Ayman, M. Sharifkhani, and M. I. Elmasry. On the design of low power mcml based ring oscillators. In *Proc. Canadian conference on Electrical and Computer Engineering*, pages 2383–2386, 2004.

[6] E.M. Camacho-Galeano, C. Galup-Montoro, and M.C. Schneider. A 2-nw 1.1-v self-biased current reference in cmos technology. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 52(2):61 – 65, feb. 2005.

[7] F. Cannillo, C. Toumazou, and T. S. Lande. Nanopower subthreshold mcml in sub-micrometer cmos technology. *IEEE Transactions on Circuits and Systems –I: Regular Papers*, 56(8):1598–1611, 2009.

[8] Z. Cashero, A. Chen, R. Hoppal, and T. Chen. Fast evaluation of analog circuits using linear programming. In *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on*, pages 253 –258, july 2010.

[9] A. Chacón-Rodríguez. *Circuitos integrados de bajo consumo para deteccion y localizacion de disparos de armas de fuego*. PhD thesis, Facultad de Ingenieria, Departamento de Ingenieria Electronica, Universidad Mar del Plata, Mar del Plata, Argentina, May 2009.

[10] A. Chacón-Rodríguez, P. Julian, L. Castro, P. Alvarado, and N. Hernández. Evaluation of gunshot detection algorithms. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 58(2):363 –373, feb. 2011.

[11] D.W Corne and J.D Knowles. The pareto envelope-based selection algorithm for multiobjective optimization. In *PPSN VI: Proceedings of the International Conference on Parallel Problem Solving from Nature*, pages 839–848, 2000.

[12] Minor Coto. Desarrollo de una plataforma de prueba para un circuito integrado de uso específico de ultra baja potencia mediante instrumentación definida por software. Licenciatura thesis, Escuela de Ingeniería Electrónica, Tecnológico de Costa Rica, 2013.

[13] George B. Dantzig and Mukund N. Thapa. *Linear programming 1: Introduction*. Springer-Verlag, 1997.

[14] B. De Smedt and G. Gielen. Watson: Design space boundary exploration and model generation for analog and rf ic design. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 213–223, 2003.

[15] Berny Dinarte. Diseño de una fuente auto-polarizada independiente de la tensión de alimentación y la temperatura para circuitos integrados analógicos. Licenciatura thesis, Escuela de Ingeniería Electrónica, Tecnológico de Costa Rica, 2011.

[16] P. Doerfler, P. Alvarado, and K.-F. Kraiss. *Advanced Man-Machine Interaction. Fundamentals and Implementation*, chapter LTI-Lib - A C++ Open Source Computer Vision Library, pages 399–421. Signals and Communication Technology. Springer Verlag, 2006.

[17] P. Drennan and C. McAndrew. Understanding mosfet mismatch for analog design. *IEEE Journal of Solid State Circuits*, 38(3):450–456, 2003.

[18] C.C. Enz and E.A. Vitoz. *Designing Low Power Digital Systems: Emerging Technologies*, chapter 1.2 CMOS Low-Power Analog Circuit Design, pages 79–133. Hindawi, 1996.

[19] Quagliarella et al. *Genetic Algorithm and Evolution Strategy in Engineering and Computer Science. Recent Advances and Aplications*. John Wiley and sons, West Sussex, England, 1998.

[20] M. R. Everingham, H. Muller, and B. T. Thomas. Evaluating image segmentation algorithms using the Pareto Front. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Proceedings of the 7th European Conference on Computer Vision*, volume IV of *LNCS 2353*, pages 34–48. Springer, June 2002.

[21] C. Galup-Montoro, M.C. Schneider, A.I.A. Cunha, F.R. de Sousa, H. Klimach, and O.F. Siebel. The advanced compact mosfet (acm) model for circuit analysis and design. In *Custom Integrated Circuits Conference, 2007. CICC '07. IEEE*, pages 519 –526, sept. 2007.

[22] O. Garitselov, S.P. Mohanty, and E. Kougianos. Fast optimization of nano-cmos mixed-signal circuits through accurate metamodeling. In *Quality Electronic Design (ISQED), 2011 12th International Symposium on*, pages 1 –6, march 2011.

[23] G. Gielen, T. McConaghy, and T. Eeckelaert. Performance space modeling for hierarchical synthesis of analog integrated circuits. In *ACM/IEEE Design Automation Conference (DAC) Circuits and Systems*, 2005.

[24] G.G.E. Gielen, H.C.C. Walscharts, and W.M.C. Sansen. Analog circuit design optimization based on symbolic simulation and simulated annealing. *Solid-State Circuits, IEEE Journal of*, 25(3):707 –713, jun 1990.

[25] M. Gilli and P. A. Winker. A review of heuristic optimization methods in econometrics. *Swiss Finance Institute Research*, 12(8), 2008.

[26] H. Graeb, D. Mueller, and U. Schlichtmann. Pareto optimization of analog circuits considering variability. In *European Conference on Circuit Theory and Design (ECCTD)*, 2007.

[27] H. Graeb, S. Zizala, J. Eckmueller, and K. Antreich. The sizing rules method for analog integrated circuit design. In *Computer Aided Design, 2001. ICCAD 2001. IEEE/ACM International Conference on*, pages 343 –349, 2001.

[28] R. Harjani, R. Rutenbar, and L. Carley. Oasys: A framework for analog circuit synthesis. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1247–1266, 1989.

[29] H. Hassan, M. Anis, and M. Elmasry. Mos current mode circuits: analysis, design, and variability. *IEEE Transactions on VLSI Systems*, 13(8):885–898, 2005.

[30] M.delM. Hershenson, S.P. Boyd, and T.H. Lee. Optimal design of a cmos op-amp via geometric programming. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, pages 1 –21, jan 2001.

[31] José Andrés Ibarra. Diseño de un filtro analógico para la detección de disparos de armas de fuego usando amplificadores operacionales de transconductancia. Licenciatura thesis, Escuela de Ingeniería Electrónica, Tecnológico de Costa Rica, 2011.

[32] National Instruments. Instrumentación virtual e instrumentación tradicional [online]. 2014 [visitado el May 17, 2014]. URL http://www.ni.com/white-paper/4757/es/.

[33] B.K. Khan and Y.A. Khalifa. An evolutionary method for analog circuits optimization utilizing mosfet-c filters. In *Electronics, Circuits and Systems (ICECS), 2011 18th IEEE International Conference on*, pages 362–365, Dec 2011.

[34] S. Kirkpatrick, C.D. Gelett, and M.P. Vecchi. Optimization by simulated annealing. *Science*, (220):621–630, 1983.

[35] J. D. Knowles and D. W. Corne. Local search, multiobjective optimisation and the Pareto archived evolution strategy. In *Proc. of the third Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems*, pages 209–216, 1999.

[36] L. A. MacEachern. Constrained circuit optimization via library table genetic algorithms. In *Proc. IEEE Int. Symp. Circuits and Systems ISCAS '99*, volume 6, pages 310–313, 1999.

[37] F. Medeiro-Hidalgo, R. Dominguez-Castro, A. Rodriguez-Vazquez, and J.L. Huertas. A prototype tool for optimum analog sizing using simulated annealing. In *Circuits and Systems, 1992. ISCAS '92. Proceedings., 1992 IEEE International Symposium on*, volume 4, pages 1933 –1936 vol.4, may 1992.

[38] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Chem. Phys., Journal of*, (21):1087 –1092, 1953.

[39] M. Mizuno. A GHz mos, adaptive pipeline technique using mos current-mode logic. *IEEE Journal of Solid State Circuits*, 31(6):784–791, 1996.

[40] D. Müller-Gritschneder. *Deterministic Performance Space Exploration of Analog Integrated Circuits considering Process Variations and Operating Conditions*. PhD thesis, Technical University Munich. Munich, Germany, June 2009.

[41] O. Musa and M. Shams. An efficient delay model model for mos current-mode logic automated design and optimization. *IEEE Transactions on Circuits and Systems –I: Regular Papers*, 57(8):2041–2052, 2010.

[42] J. M. Musicer and J. Rabaey. MOS Current mode logic for low power, low noise, CORDIC computation in mixed-signal environments. In *Proc. IEEE Int. Symp. on Low power electronics and design*, pages 102–107, 2000.

[43] Frank Nicaragua. Diseño de un amplificador operacional de transconductancia para la implementación de filtros analógicos utilizados en la detección de disparos de armas de fuego. Licenciatura thesis, Escuela de Ingeniería Electrónica, Tecnológico de Costa Rica, 2010.

[44] S. Obayashi. Multidisciplinary design optimization of aircraft wing planform based on evolutionary algorithms. In *IEEE International Conference on Systems, Man, and Cybernetics*, 1998.

[45] M.J.M. Pelgrom, Aad C J Duinmaijer, and A.P.G. Welbers. Matching properties of mos transistors. *Solid-State Circuits, IEEE Journal of*, 24(5):1433–1439, Oct 1989.

[46] M.J.M. Pelgrom, H.P. Tuinhout, and M. Vertregt. Transistor matching in analog cmos applications. In *Electron Devices Meeting, 1998. IEDM '98. Technical Digest., International*, pages 915 –918, dec 1998.

[47] R. Pereira-Arroyo, P. Alvarado-Moya, and W. H. Krautschneider. Design of a mcml gate library applying multiobjective optimization. In *Proc. IEEE Computer Society Annual Symp. VLSI ISVLSI '07*, pages 81–85, 2007.

[48] R. Pereira-Arroyo, F. Nicaragua-Guzmán, and A. Chacón-Rodríguez. Design of an operational transconductance amplifier applying multiobjective optimization. In *Argentine School of Micro-Nanoelectronics Technology and Applications (EAMTA), 2010*, 2010.

[49] Sadiq M. Sait and Youssef Habib. *VLSI Physical Design Automation: Theory and Practice*. World Scientific Publishing, 1995.

[50] G. Stehr. *On the Performance Space Exploration of Analog Integrated Circuits*. PhD thesis, Technical University Munich. Munich, Germany, Sept. 2005.

[51] R.A. Vural, T. Yildirim, T. Kadioglu, and A. Basargan. Performance evaluation of evolutionary algorithms for optimal filter design. *Evolutionary Computation, IEEE Transactions on*, 16(1):135–147, Feb 2012.

[52] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transaction on Evolutionary Computation*, 3(4):257–271, November 1999.

[53] A. Zjajo, Qin Tang, M. Berkelaar, J.P. de Gyvez, A. Di Bucchianico, and N. van der Meijs. Stochastic analysis of deep-submicrometer cmos process for reliable circuits designs. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 58(1):164–175, 2011.

Appendix A

Optimization tool user's manual

This appendix gives a description of the process that needs to followed in order to set up an optimization environment for a given circuit, such as any of the ones presented in this dissertation. The first requirement is that a schematic simulation of the circuit is set up in a commercial simulator. In this case, *Mentor Graphics* has been used and this discussion will be confined to it.

The tool basically needs to directories, usually created at *home* directory:

1. *simInterface*: it contains a set of *C++* routines that link simulation files and the optimization algorithm. Computation of fitness values takes place here.
2. *ltilib-pareto*: it is an onject-oriented library also in *C++*, where the genetic algorithm is implemented and the Pareto front is generated.

Working directories should be created within both *simInterface* and *ltilib-pareto*, as for example:

```
\home\user\simInterface\MI_OPTIMIZACION
\home\user\ltilib-pareto\examples\MI_OPTIMIZACION
```

A.1 Circuits' Parameter Definition

In order to parameterize a schematic, numeric values must be substituted by variables. This is accomplished in *Mentor Graphics* as follows:

1. Go to component properties (right click > *Properties* > *Edit*).*Properties* can also be reached if the component is selected and press letter *q* (Caps Lock must NOT be active).
2. Select the desired parameter and in the field *Value*, enter the letter you wish that parameter to be identified with. In the field *Type*, select option *String* and confirm by

pressing *Apply* each time a parameter is modified. Finally, click *OK* to finish editing the component. Fig. A.1 shows the process just described in order to parameterize the length of a transistor.

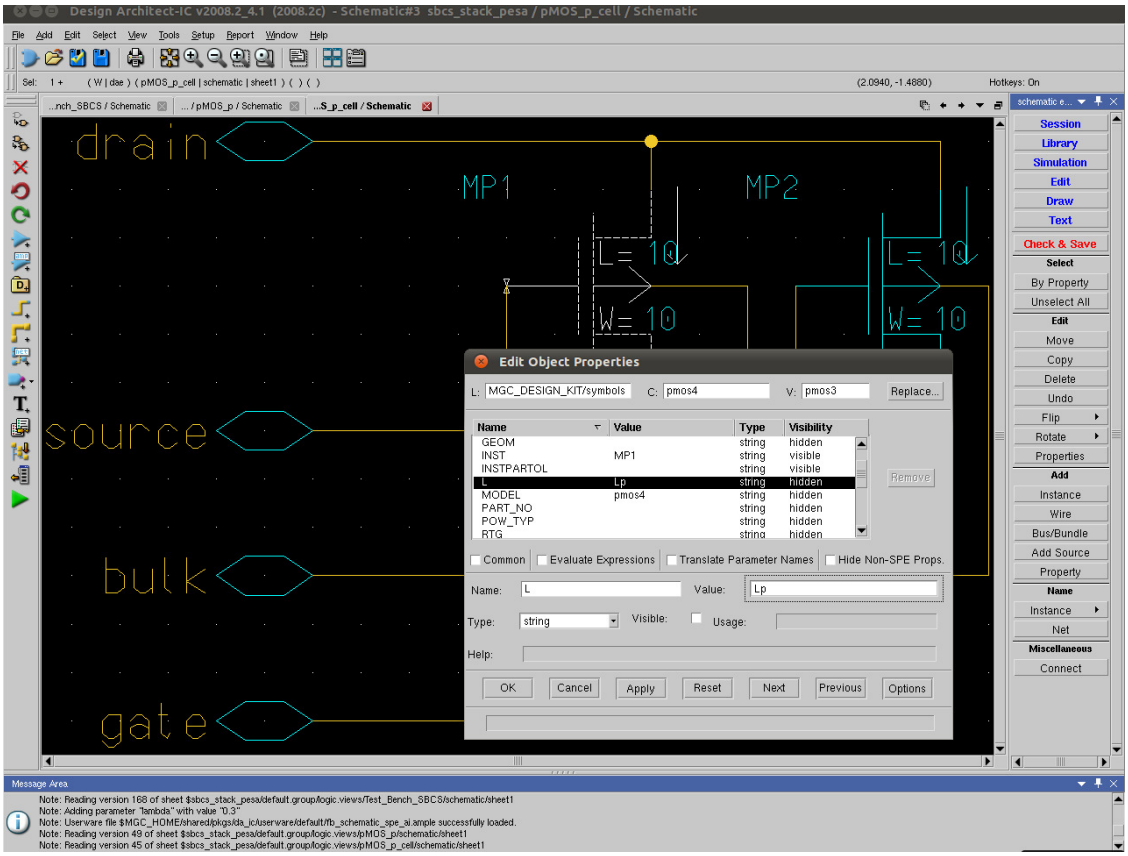


Figure A.1: Process for the parameterization of variables.

3. Repeat this procedure for all parameters that need to be changed during the optimization process (transistor dimensions, current and voltage values, resistors, capacitors, etc.). Keep track of which parameters have been defined as variables for later use

4. Go to the simulation environment (in the schematic window, get into *Simulation* mode). Then goto *Multiple Runs* tab, *Sweep* option. Place there the defined variables and register numeric values such that the simulator can be executed. Fig. A.2 shows the definition of parameters for a specific design.

5. Still inside the simulation environment, get into *Options* tab. Choose the optio *Standard* and then the *Outp(2)* tab, such that the option *SPI3ASC* is enabled in order thet files generated by *Eldo Spice de Mentor Graphics* have *ASCII* format.

6. Verify that the simulation is working fine. Find the output file generated by *Eldo Spice de Mentor Graphics*. These are normally located at:

```
\home\user\mentor\proyecto.proj\biblioteca.lib\default.group
\logic.views\celda\configuracion_simulacion\
```

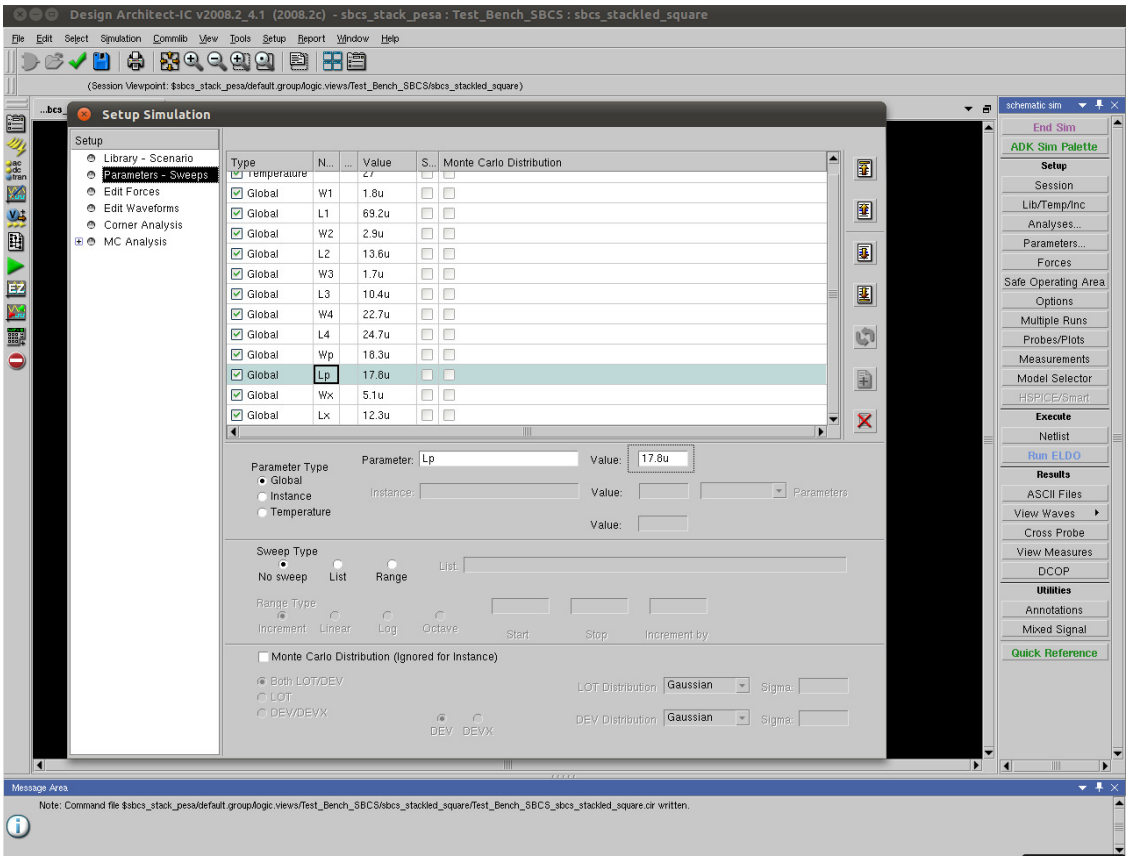


Figure A.2: Example of parameter definition.

Copy the files with extensions *.chi*, *.cir*, *.spi* y *.spi3* into the following directory:

```
\home\user\simInterface\MI_OPTIMIZACION
```

7. The implementation of the tool requires that the file with extension *.cir* and copied in directory *simInterface* be modified. To do so, open a text editor and proceed as follows:

   7.1. Modify the second line of the file, where the *.INCLUDE* is located. Get rid of the path.

   ```
   .INCLUDE archivo.spi
   ```

   7.2. Put all parameters into a single line *.PARAM* as shown here:

   ```
   .PARAM L1=8.67U L2=4U L3=3.8U
   ```

Appendix B

ltilib-pareto

In the directory:

```
\home\user\ltilib-pareto\examples\MI_OPTIMIZACION
```

The following file must be modified:

cmlFunctor.h

Here the parameters are defined.

1. Define the circuit parameters, using the same names as those given in the Netlist. Use float type.
2. Edit their names in the class *paretoFrontTester* such that they are compatible with the defined parameters. Modify the instances bitsFor-name_of_parameter. More lines can be added if needed. Currently 8 bits are used for every parameter but this can be adjusted.

cmlFunctor.cpp

1. Edit const int paretoFrontTester::totalBits, such that it matches instances bitsFor.. previously created.
2. Edit constructor cmlFunctor with the parameters that will be used.
3. Edit other instructions that employ parameter names.
4. Edit write and read handlers de escritura y de lectura, again to match with circuit parameters

5. Define parameter range by setting the minimum and maximum values for every parameter.

6. Edit the instructions pos=... with the variables previously defined.

7. Asign a socket number.

8. Edit the command sstr. This line, when executed, writes back into the netlist. Therefore it must be defined in exactly the same order as in the Netlist.

9. The command ist reads fitness data coming from the simulator interface. Take a look at it and check the order in which these come.

circuitEvaluatorMain.cpp

This file only requires instructions to be adjusted to the amount of parameters defined.

config_cmleval.dat

This file can be modified in order to set specific values to the genetic algorithm. It is compulsory to adjust the option (fitnessSpaceDimensionality) to the amount of defined fitness functions. Option (numOfIterations 2500) can be adjusted depending on simulation run time (aproximately 3 seconds per iteration and 10 simulations per iteration).

B.1 simInterface

The files with extensions *.chi*, *.cir*, *.spi* y *.spi3* are located in the following folder:

```
\home\user\ltilib-pareto\examples\MI_OPTIMIZACION
```

start

Modify the *start* file as follows:

1. The command cd must point to *MI_OPTIMIZACION*, within simInterface.

2. The command source must point to bash_me_adk. This is a script required to configure Mentor variables.

3. The command eldo must call the file with .cir extension that is located in simInterface.

```
#!/bin/bash
cd /home/user/simInterface/MI_OPTIMIZACION
source /home/user/mentor/bash_me_adk
eldo /home/user/simInterface/MI_OPTIMIZACION/nombre_archivo.cir
```

IOGenetic.cpp

1. The array parameters[] contains initial values for every one of the defined circuit parameters. Always keep the order in which they appear.

2. The size of the array Fitns[] must be adjusted to match the number of fitness functionsdefined. As an advice put the function that you consider the most important as Fitns[0], but otherwise they can be defined arbitrarily in any position of the array.

3. Define the needed arrays for all data coming from the output file *.spi3*. This files contains a collection of samples for every type of analisys (dc, ac, transient, etc.) set up in the simulation.

4. Within functions write_parameters() and read_parameters(), modify fstream such that its argument contains the path to the *.cir* file:

   ```
   /home/user/simInterface/MI_OPTIMIZACION/nombre_archivo.cir
   ```

   Write parameters simply overwrites in each call the line *.PARAM* with data coming from the optimizer

5. The method read_parameters() must be adjusted to the amount of information given by the simulation in the .spi3 file.

6. The method FitGen() computes the fitness values that were defined by the user.

7. The following modification must be carried out on the main() method:

   7.1. Define the port number in the instruction ServerSocket server (# de puerto), matching the port number assigned in the file cmlFunctor.cpp in ltilib-pareto.

   7.2. In the definition of int comma, use the whole path downto the file start

   7.3. Instruction sstr< creates a string with fitness values and an ID tag for every one, if this ID has more than one word in it, then use an underscore to make it a single string.(For example, use slew_rate instead of slew rate).

8. Make sure all routines work correctly before starting the optimization. In the main(), comment anything that has to do with running the optimization when doing these preliminary tests, as shown below.

```
 int main () {

   std::cout << "running....\n";
/*INICIA COMENTARIO
     try
         {
       // Create the socket
             ServerSocket server ( 11050);
      ...
      ...
      ...
       catch ( SocketException& e ){
       std::cout << "Exception was caught:" << e.description() << "\nExiting.\n";
}
*/FINALIZA COMENTARIO

//PROBANDO METODOS
readparameters ("Values:")
return 0;

}
```

Appendix C

Execution of the tool

It is recommended to start with short optimization runs in order to check correct operation of all adjustments made within the simInterface folder.

1. Open a terminal and go to this directory:

   ```
   \home\user\simInterface\MI_OPTIMIZACION
   ```

   Execute:

   ```
   make clean
   make
   ```

   Solve errors if any.

2. Open another terminal and go to this directory:

   ```
   \home\user\ltilib-pareto\examples\MI_OPTIMIZACION
   ```

   Execute:

   ```
   rm pareto.log ; rm pareto.log.pf
   make clean
   make
   ```

   Solve errors if any.

3. For the terminal opened under simInterface, execute:

   ```
   ./simple_server
   ```

This open the server that links the simulator with the client, i.e. the genetic algorithm and Pareto front generator. If you want to save the standard output, then type for example:

```
./simple_server > salida_estandar
```

4. For the terminal opened under ltilib-pareto, execute:

   ```
   ./MI_OPTIMIZACION > resultados
   ```

   The name of the executable is the same as the directory, in this case *MI_OPTIMIZACION*. With this command the optimization process is initiated. The results are stored in the file named *resultados* (of course any other name could be used!). This file will contain a list of the parameters and their corresponding fitness values.

5. Open a terminal under ltilib-pareto and execute:

   ```
   tail -f resultados
   ```

   This will allow seeing the last 10 lines of the file *resultados* such that you can take a look at the progress of the optimization.

6. When the terminal opened at the previous item shows:

   ```
   It seems we're at the end that's all folks
   Test at the end!
   ```

   it means the optimization has finished. To return to the promt press *Ctrl + C*).

7. The folder *MI_OPTIMIZACION* located at ltilib-pareto, contains the files *pareto.log*, *pareto.log.pf* and *resultados*. Bear in mind that this files have to be saved apart for later use.

Appendix D

Processing of results

After the optimization run is finished, the point or points which satisfy the desired requirements must me chosen from the Pareto front. Revisiting the example of section 3.3, the point with minimum P, minimum t_{ph} and maximum f_c will be searched for. It must be recalled that the optimization tool works with increasing fitness values. Therefore, P and t_{ph} are inverted. A segment of the Pareto front of the aforementioned example looks like:

```
 (data (62855500000 4558140000 0.0256838))))
(((size 3)
  (data (62855500000 4558140000 0.0256838))))
(((size 3)
  (data (62694400000 4635570000 0.025858))))
(((size 3)
  (data (62694400000 4635570000 0.025858))))
(((size 3)
  (data (62685700000 4636490000 0.0258737))))
(((size 3)
  (data (62684100000 4636520000 0.025876))))
```

Take, for example, the point (62685700000 4636490000 0.0258737), as a point that satisfies the requirements previously imposed. Then the next step is to identify which parameterization originated that point. This is achieved by examining the file *resultados*, i.e. the file that contains the standard output, as explained in appendix C. The result of such a search is shown below:

```
               Internal evaluation 10/10
Valores de W1: 0.8
Valores de L1: 45.6706
Valores de W2: 22.5569
Valores de L2: 3.12941
Valores de W3: 1.02902
```

```
Valores de L3: 19.2392
Valores de W4: 1.60157
Valores de L4: 3.31765
Valores de Wp: 0.8
Valores de Lp: 23.6314
Valores de Wx: 1.25804
Valores de Lx: 17.9216
We received this response from the server:
"Iout: 6.26857e+10 Power: 4.63649e+09 %/V: 0.0258737"
Valor fitness0: 6.26857e+10
Valor fitness1: 4.63649e+09
Valor fitness2: 0.0258737
```

It is recommended to simulate again the found parameterization to double check that the circuit still works fine.

Appendix E

Generation of the Pareto front graph

In order to generate the graph of a Pareto front copy the file *pareto.log.pf* into this directory:

```
/home/user/ltilib-pareto/examples/pareto
```

Take a close look at the *README* file in this directory. It gives the rest of the steps in order to generate the graph.

```
This example creates GNU-Plot files for the output of the examples using a
lti::paretoFront derived class (see example cwagmeval).

It can generate graphics for 2D and 3D Pareto Fronts.

After compiling (just execute "make") you need a data file (pareto.log,
pareto.log.pf or similar) for which the front will be generated.

As an example:  if you executed the cwagmeval example then just do

> pareto -p test.gp -x ../cwagmeval/pareto.log.pf

This will create a GNU-Plot script test.gp that can be executed with

> gnuplot -persist test.gp

  (you can close the window presing 'q')

You can edit test.gp to fit your need, and as a hint, if you replace -x with
-f, the generated script will instruct gnuplot to export an xfig file or with
-e an eps file.

You can specify several files at the same time, so that different fronts are
```

plotted on the same graphic. With -a you can also get the front of all fronts.

For a Pareto 3D graph proceed as follows:

```
./pareto -p script_frente.gp -e -n -3 pareto_fecha_version.log.pf
gnuplot -persist script_frente.gp
```

Fig. E.1 shows an example of such a front.

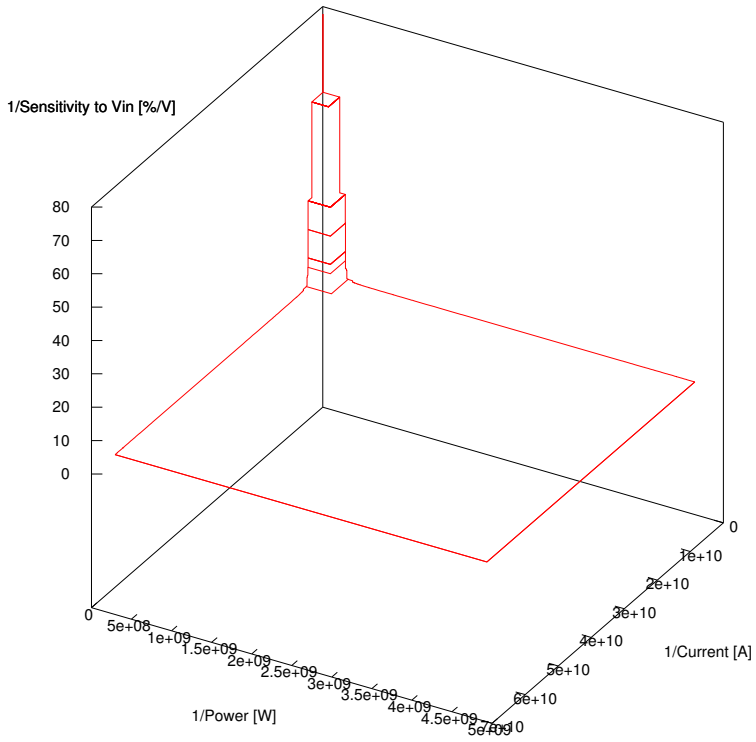


Figure E.1: Example of a generated Pareto graph in 3D.

Index