UNIVERSITY OF SOUTHAMPTON

System Level Performance and Yield Optimisation for Analogue Integrated Circuits

by

Sawal Hamid Md Ali

A thesis submitted for the degree of

Doctor of Philosophy

in the

Faculty of Engineering, Science and Mathematics

School of Electronics and Computer Science

November 2009

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINNEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Sawal Hamid Md Ali

Advances in silicon technology over the last decade have led to increased integration of analogue and digital functional blocks onto the same single chip. In such a mixed signal environment, the analogue circuits must use the same process technology as their digital neighbours. With reducing transistor sizes, the impact of process variations on analogue design has become prominent and can lead to circuit performance falling below specification and hence reducing the yield.

This thesis explores the methodology and algorithms for an analogue integrated circuit automation tool that optimizes performance and yield. The trade-offs between performance and yield are analysed using a combination of an evolutionary algorithm and Monte Carlo simulation. Through the integration of yield parameter into the optimisation process, the trade off between the performance functions can be better treated that able to produce a higher yield. The results obtained from the performance and variation exploration are modelled behaviourally using a Verilog-A language. The model has been verified with transistor level simulation and a silicon prototype.

For a large analogue system, the circuit is commonly broken down into its constituent sub-blocks, a process known as hierarchical design. The use of hierarchical-based design and optimisation simplifies the design task and accelerates the design flow by encouraging design reuse.

A new approach for system level yield optimisation using a hierarchical-based design is proposed and developed. The approach combines Multi-Objective Bottom Up (MUBU) modelling technique to model the circuit performance and variation and Top Down Constraint Design (TDCD) technique for the complete system level design. The proposed method has been used to design a 7^{th} order low pass filter and a charge pump phase locked loop system. The results have been verified with transistor level simulations and suggest that an accurate system level performance and yield prediction can be achieved with the proposed methodology.

DECLARATION OF AUTHORSHIP

I, Sawal Hamid Md Ali, declare that the thesis entitled System Level Performance and Yield Optimisation for Analogue Integrated Ciruits and the work presented in it are my own. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledge all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as:
    - Sawal Ali, R. Wilcock and P.R. Wilson, "Improved performance and variation modelling for hierarchical-based optimisation of analogue integrated circuits", *Design, Automation and Test in Europe (DATE) 2009, Nice, France.*
    - Sawal Ali, R. Wilcock, P.R. Wilson and A.D. Brown, "A new approach for combining yield and performance in behavioural model for analogue integrated circuits", *Design, Automation and Test in Europe (DATE) 2008, March 3-7, Munich.*

Signed: ....

Date: ....

Acknowledgements

Alhamdulillah, I thank to Allah for giving me a good health and the opportunity to continue my life and finish my study successfully.

I wish to express my sincere gratitude to my supervisor, Dr. Peter R. Wilson for providing the opportunity to carry out this study, and for guidance and support. I am deeply indebted to Dr. Reuben Wilcock for his valuable suggestion and active cooperation and having been a part of every single stage of my thesis, from inception to completion.

I wish to extend my warmest thank to all the staff in the Electronics System Devices Group, School of Electronics and Computer Science who have helped me in my work throughout my study.

Collective and individual acknowledgments are also owed to my colleagues at ESD group whose present somehow perpetually refreshed, helpful and memorable especially to Noohul Basheer, Abu Bakar, Sankalp Modi, Karthik Badam, Arash Ahmadi, Biswajit Misra, Li Ke, John Goodwin, Julian Bailey, for giving me such a pleasant time and valuable advice.

I convey my special acknowledgements to the National University of Malaysia (UKM), Ministry of Higher Education of Malaysia and Malaysian government for the financial support given to me and my family. I gratefully thank Tun Dr. Mahathir Mohamad, the ex-prime minister of Malaysia for being a role model for hard working person and for all his encouragements towards hardworking ethics.

My sincere thanks are given to my parents, Md Ali bin Daud and Sabura abd Karim for their inseparable support and prayers. I would like to offer my greatest appreciation to my wife Siti Anom and my little prince and princess, Syafique Azra'ei and Syra Arisha for being understanding and always stand by me with love and prayers. Thank you very much for letting me do this "geek" thing and I'm glad we're travelling together through this journey called life! I love all of you.

List of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Integrated Circuits

In 1965, Gordon Moore predicted that the number of transistors on a chip will double about every two years [1]. This statement also implies that the density of a single chip will increase due to the higher number of transistors integrated. Since then, the field of electronics had seen a huge development that has revolutionised many aspects of consumer electronics. Moving from a small number of transistors to multi million transistor circuits has provided the functionality that past generations could only dream of. Figure 1-1 shows the trend in transistor complexity for microprocessors that follow the Moore's law prediction.

One of the main reason for this prediction continue to be valid is the continuous development in transistor size reduction. This trend allows the integration of several functional blocks that previously occupied one or more boards onto a single chip, a technique that is termed as System-On-Chip (SoC). Although most of the functional blocks in an integrated system are digital, analogue circuits are still needed to interface to the real world which drives to the integration of analogue and digital circuits in a single system known generally as mixed-signal. This integration is very attractive due to the significant reduction that can be made to the device size and hence to the overall cost of the system.

One of the most important applications of analogue circuits is to bridge the gap between the `real' world and the digital domain. The need to go from analogue to digital processing have made the use of analogue-to-digital and digital-to-analogue converters indispensable. Several other important analogue components include filters, amplifiers, integrators and reference circuits for biasing. All these components are found in various applications such as communication systems, signal processors and radio frequency (RF) circuits. It is thus clear that analogue circuit integration is important and necessary in a large range of applications especially when considering SoCs where the link between the analogue and the digital domain will be required in practically every circuit.

With the rising level of integration, the complexity and the challenges of the integrated circuits increases. Such complexity has increased the requirement to use CAD tools for design automation that supports the design on several hierarchy of abstractions. The following section will discuss some of the challenges faced by the analogue circuits. This discussion will lead to the motivation behind the research that is to explore a methodology that can be used for automating and optimising the design flow of analogue circuits.

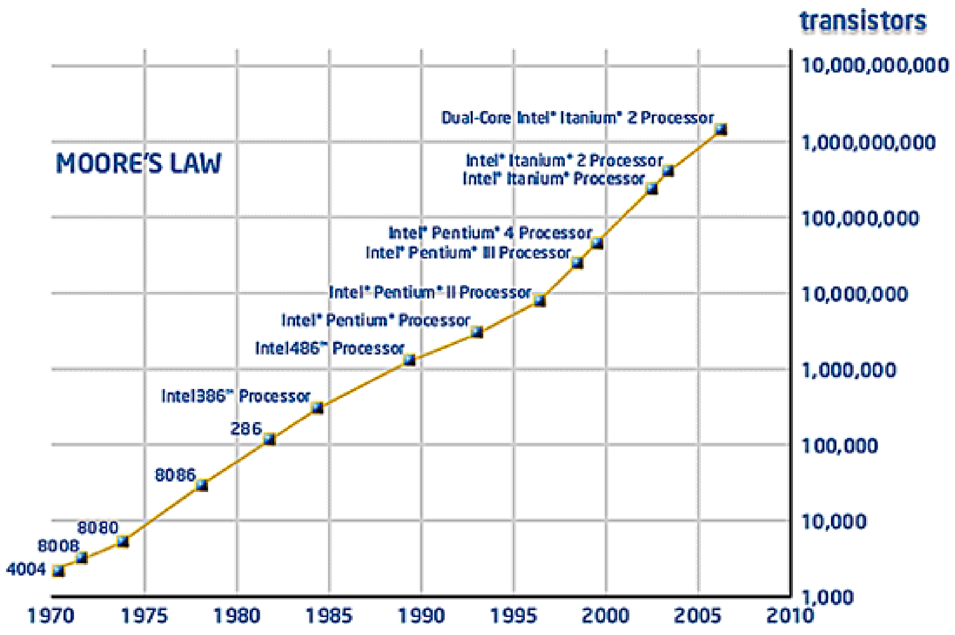


Figure 1-1: Transistor Complexity's Trend – Moore's Law

1.2 Challenges in analogue design

In a complex mixed-signal system, the analogue circuit may occupy a small area compared to the digital circuit but the design time of the analogue circuit is often much longer and can therefore cause a bottleneck in the overall system design [2]. The reasons for this are generally the circuit complexity and the lack of automation tools that can speed up the design process. Unlike digital circuits which can be rapidly synthesized by computer-aided-design tool, most of the analogue circuits are still essentially designed manually.

Another challenge faced by the analogue circuit in a mixed-signal environment is often the requirement to use the same transistor process technology as the digital circuits. For digital circuits, process technology downscaling is desirable due to the capability to reduce power consumption, area and delay. However, this is not necessarily helpful for analogue circuits. For example, a reduction in supply voltage due to the small transistor size, limits the voltage swing of the signals in the circuit and this can increase the signal to noise ratio and total harmonic distortion of the circuit. This has proved to be a significant challenge to analogue circuit designers in term of optimising the design for better performances and meeting the specifications.

Furthermore, as the transistor sizes are scaled down, the resulting variability increases and adversely effect yield. These variations in the process technology have a large influence to the quality and yield of a designed and manufactured circuit. With further shrinking of process technology, the variation is getting worse for each technology node. For technologies larger than 180nm feature sizes, variations are mostly in a range of below 10%. However, shrinking technologies down to 90nm, 65nm and below cause the variations to be more than 50% [3]. With a high correlation of circuit yield to profit, yield maximisation has became a major issue in deep sub-micron integrated circuit design and has been considered as an important factor in the design stages.

This thesis addresses one of the important topics in analogue IC design, which is to optimise the performance and yield of deep submicron integrated circuit design. The method proposed in the thesis starts with performance and variation model development using a Pareto front approach and is followed by a top-down system design methodology using a hierarchical flow, that provides the designer with the ability to optimise the design for better performance and higher yield at the system level.

1.3 Project motivation and goal

The difficulties in the design of analogue integrated circuit (IC) discussed earlier shows some of the challenges faced by the analogue designer. Increase of design complexity, impact of process variations and demand for design cycle time reduction increase the need to have a new improved methodology for analogue design automation tool. Recent advances in design automation have led to a gradual transition from “hand-calculation” based design to a simulation-based sizing methodology [4]. A Simulation-based approach tests many circuit candidates during the sizing process and evaluates each candidate via detailed circuit simulations. For a large circuit, the searching space for optimization can be very large and this increases the simulation time significantly. One of the solutions to this problem is modelling the performance space of the circuits behaviourally such that the optimisation can be done without the need of repeating extensive circuit simulation, at a transistor level.

In addition, the higher impact of process variation on the design yield has led to the integration of a yield parameter as one of the performance parameters in the design process. Although there is extensive research in this area, most do not model the performance variation together with their performance model and hence has no ability to predict the yield directly. Most of the current methods exist in yield optimised design are based on an approximation model and only focus at circuit level optimisation [5, 6, 7, 8]. The methodology presented in this thesis focuses on performance and variation modelling, and a top-down hierarchical design technique that is suitable for performance and yield optimisation for both at circuit level and system level design. The specific objectives of this project are discussed in the remainder of this chapter.

1.4 Project Scope

1.4.1 Introduction

The scope of this project is to develop the ideas for modelling circuit performance and their variation that can be used efficiently and accurately in the design of analogue integrated circuits.

Specifically, the project involves several activities including:-

- Parameter extraction that relates the circuit performances and their design parameters.
- Yield characterizing that relates the performances and their variations through a minimum and maximum estimation from a Monte Carlo simulation.
- Construction of behavioural model of a circuit example to model the performance and variation.
- Hierarchical-based optimisation design flow for system level design, and
- Methodology verification with practical examples.

When considering a performance and variation model of an analogue circuit, one of the most important factors is the accuracy of the model. Often a trade-off is being made that trades the accuracy for speed of simulation. In this thesis, the accuracy of the model is given a high weighting and the technique chosen for the model development reflects this intention. Several examples have been chosen to demonstrate the model application that includes a complete design flow from design specifications through to silicon implementation.

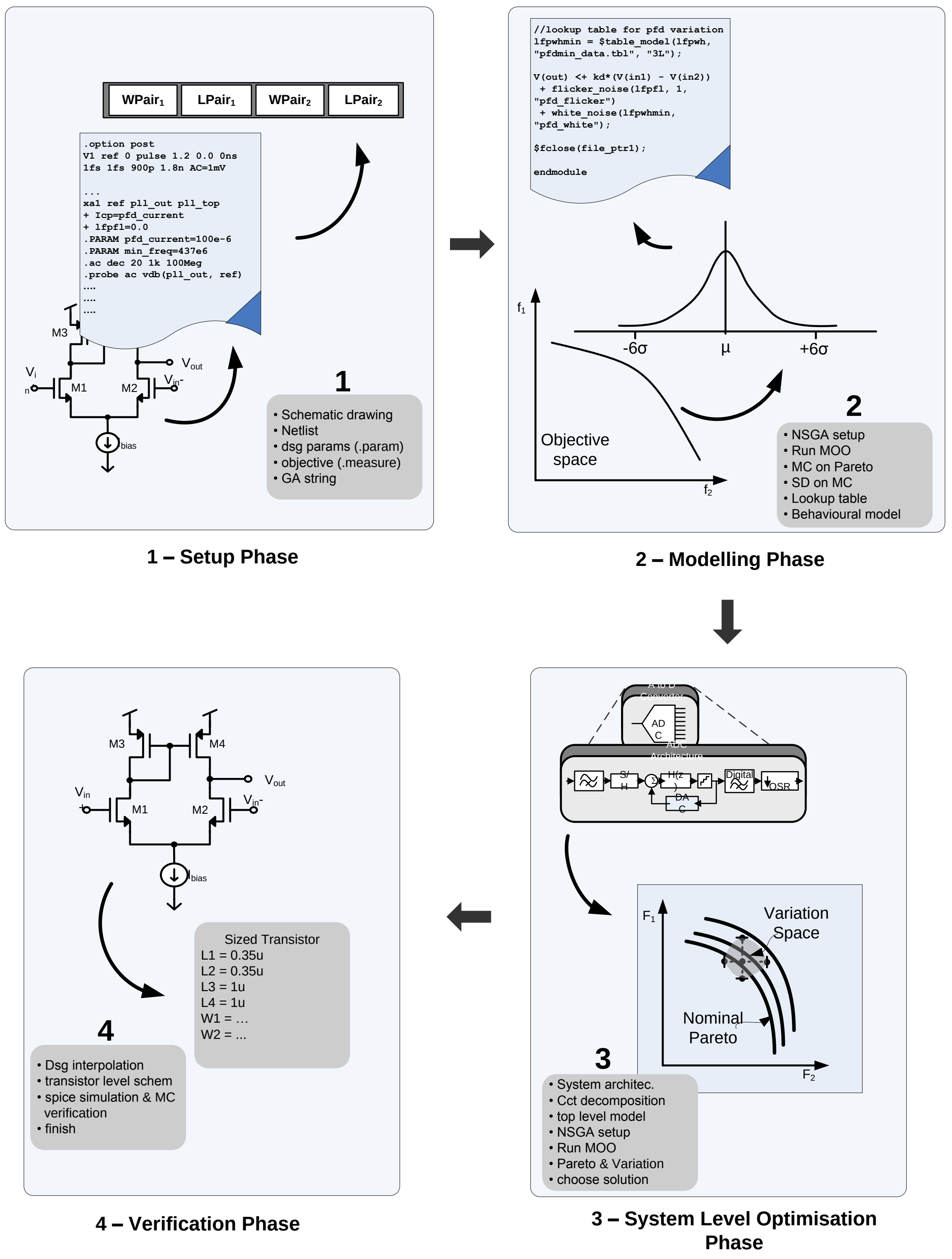
1.4.2 Structure of the Project

The project was split into three main phases and can be illustrated as shown in figure 1-2 :-

Phase 1: To establish the methods for modelling the performance and variations of a circuit design. This involves extensive review of analogue synthesis techniques and yield optimization methodologies. The transition of design automation and techniques from hand-calculation based to simulation-based was carefully studied in order to choose the suitable and accurate method for the synthesis technique. Comparison was made with other methods especially for yield optimisation technique including design centring methods and the use of commercial optimisation tools.

Phase 2: To build the performance and variation model of an example circuit design. This model was built from optimal performance points of the objective space and their minimum and maximum variation estimation based on a 6th standard deviation range. Both of the performance and variation model were developed behaviourally making it suitable for fast behavioural level simulation. A silicon prototype of a 2nd order filter was developed to demonstrate the practicality of the model and to validate the proposed methodology.

Phase 3: To develop a new hierarchical-based design technique that can be used for system level design. The performance and variation model developed in previous phase was used in the hierarchical design flow to design and optimise a system level block for performance and yield. A mixed-signal charge pump phase locked loop was used to demonstrate the full bottom up and top down design flow of the system for performance and yield optimisation.



```
.option post
V1 ref 0 pulse 1.2 0.0 0ns
1fs 1fs 900p 1.8n AC=1mV

...
xa1 ref pll_out pll_top
+ Icp=pfd_current
+ lfpfl=0.0
.PARAM pfd_current=100e-6
.PARAM min_freq=437e6
.ac dec 20 1k 100Meg
.probe ac vdb(pll_out, ref)
....
....
....
```

```
//lookup table for pfd variation
lfpwhmin = $table_model(lfpwh,
"pfdmin_data.tbl", "3L");

V(out) <+ kd*(V(in1) - V(in2))
 + flicker_noise(lfpfl, 1,
"pfd_flicker")
 + white_noise(lfpwhmin,
"pfd_white");

$fclose(file_ptr1);

endmodule
```

Figure 1-2: Project structure

1.4.3 Project Hypotheses

As a basis for the research in this project, several specific hypotheses were made as follows:-

- Existing yield optimised design methodologies have several inadequacies including the ability to predict and optimise the yield at system level design.
- In deep sub-micron technology, where the design complexity and variability has become a significant challenge, the accuracy and the ability to translate the simulated results into a real design is very important.
- Existing approaches for system level design using a hierarchical-based optimisation method do not consider the variations of the sub-block circuits leaving the yield optimisation for the system until the end of the design flow.
- A new hierarchical-based optimisation is needed that can incorporate the performance and variation model of analogue circuits into a top down system level design flow.
- The application of behavioural modelling languages such as Verilog-A allow the ability to model a system that include mixed-signal blocks and offers a huge potential saving in terms of simulation time.

1.5 Thesis Structure

This section explains briefly the main points of each chapter in the thesis. The first part of the thesis, chapters 1-3 contain the background theory and literature review which leads to chapter 4 & 5 describing the implementation of the performance and variation model for analogue circuits. The last part of the thesis investigates a demonstrator application using a proposed hierarchical-based optimisation for mixed-signal system level design. This is covered in chapter 6 and 7. Chapter 8 concludes the project and recommends areas for the future work.

1.5.1 Chapter 1: Introduction

The introduction of the thesis describes the motivations and goals to the project. The challenges in analogue circuit design are briefly explained which define the research landscape for the project.

1.5.2 Chapter 2: Review of Analogue circuit Design and Statistical Design Techniques

This chapter reviews the techniques and developments in analogue circuit design automation which can be divided into three main categories : Knowledge-based, analytical-based and simulation-based design. The optimisation techniques are reviewed and compared to provide initial understanding that is suitable in this project. Statistical design techniques for analogue circuit are reviewed and their limitations are defined in this chapter.

1.5.3 Chapter 3: Review of Simulation & Modelling

The aim of this chapter is to review and explain the modelling principles and techniques used for electronic circuits. Basic concepts of behavioural modelling are introduced here and the advantage given by the behavioural model in a system level design is described.

1.5.4 Chapter 4: Yield Optimised Design

This chapter demonstrates how to implement the performance and yield optimization model for analogue circuit design. The method of characterizing the performance and yield space is proposed. The concept of performance trade-offs and Pareto-front that will be used for the remainder of the thesis are introduced in this chapter. The algorithm for the optimization is discussed, with examples, and is compared with existing methodologies to demonstrate the effectiveness of the approach.

1.5.5 Chapter 5: Performance and Variation Modelling

This chapter describes how the multi-objective optimisation discussed in the previous chapter is used to model the performance and variation of a circuit design. The concept of performance and variation modelling from the objective space and Pareto-front are introduced in this chapter. A new approach for combining the performance and variation model using a lookup-table implementation in Verilog-A is proposed and the implementation with a behavioural table model function is explained. An example is used to illustrate the development of the performance and variation model and a practical example with a silicon prototype is shown for the practicality aspect of the methodology.

1.5.6 Chapter 6: Hierarchical-based Design Optimisation

This chapter describes how the performance and variation model can be used in a system level design using a hierarchical-based optimisation technique. A new modification is done to the hierarchical-based method to include both the multi-objective bottomup modelling and top-down constrained design in the algorithm. A 7th order elliptic filter for video applications is used to demonstrate the methodology.

1.5.7 Chapter 7: Mixed Signal System Level Application

In this chapter, A charge pump PLL is used as a mixed-signal system example with higher number of design parameters, objective functions and mixed domain simulations to demonstrate the effectiveness of the proposed methodology to optimise the performance and yield for significant circuit sizes.

1.5.8 Chapter 8: Conclusion and Future Work.

In this chapter, the results obtained are discussed. The accuracy of the model especially in a practical example is discussed. Conclusions are drawn from these discussions and a statement about the hypotheses is made. Finally the areas that could provide the basis of future work are highlighted.

Chapter 2

Review of Analogue Circuit Design and Optimisation

2.1 Introduction

Analogue circuit design can be divided into two main tasks: The selection of an appropriate circuit topology and circuit sizing. The design starts with a circuit specification that defines the performance functions and their upper and lower limits. Based on the specification, a topology will be selected. There is the possibility that several topologies existed, that implements the required functionality. Usually the topology selection is based on design heuristics. The knowledge or experience of the designer is often the main approach used to find the suitable topology that can meet the design requirements. The next step is to determine the size of the devices for the selected topology. This step is called circuit sizing and the parameters to be sized are called design parameters. The sizing process of design parameters will determine the performance of a circuit. This step is a complicated task due to the nonlinear relationship between the design parameters and circuit performance.

Usually the sizes of the design parameters are approximated using simplified hand-calculations. The formulas are based on simple approximations of the transistor characteristics that may differ from the real devices. The approximated circuit sizes will be used as the initial point for the performance evaluation using a circuit simulator such as HSPICE [9] or Spectre [10]. For the purpose of the simulation, a test bench is created where a set of suitable input signals are applied to the circuit in order to extract the performance functions. This will give the initial performance of the circuit and most certainly will not meet the specification. Thus, the device sizes must be adjusted through the optimisation process. Some simulators offer a simple form of optimisation to adjust the device parameters in order to fine-tune the performance functions. If no feasible solution is found during the optimisation and the specifications are not met, a different circuit topology must be selected and the sizing and optimisation processes will need to be repeated. This will eventually increase the design cycle time of the analogue circuits and becomes the bottleneck in the design process. According to [11], the design cycle time reduction can be managed only by the use of computer aided design. Therefore, over the years, the research community has been aggressively working towards the development of computer aided design tools for analogue circuits. A good survey of analogue synthesis techniques is available in [12] and will be reviewed later in this chapter.

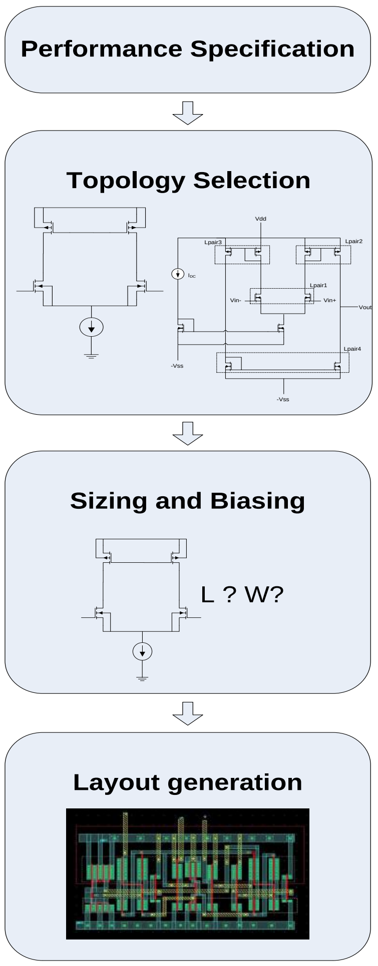


Figure 2.1 Typical design flows for analogue IC design

Figure 2.1 shows a typical design flow in analogue IC design. One of the most important aspects in the design flow are the time spent on designing the low-level cells. The time required to design an amplifier for example might be in the order of weeks [13] when all design steps are considered. Decreasing the time spent on the design process through automation techniques for instance will have a large impact on the time-to-market for the whole chip. This automation can be applied at different steps in the design flow, for example, topology selection or circuit sizing. This thesis will focus on circuit sizing automation techniques and the performance and variation models were targeted at the circuit sizing stages. The remainder of this chapter will review the existing approach for analogue circuit sizing.

2.2 Automatic Circuit Sizing

The approach in automatic circuit sizing can be classified into two main categories, namely knowledge-based design and optimization-based design. Optimization-based design can be further divided into two approaches, equation-based optimization and simulation-based optimization.

2.2.1 Knowledge-Based Design

This is one of the earlier approaches in automated circuit sizing. The basic idea is to have a predefined design plan or design rules on how to size circuit components to meet the performance specifications. The design plan generally consists of a set of design equations for a particular circuit topology. In knowledge-based design, these equations are formulated so that with a given circuit performances, the size of the circuit can be determined.

Once the design plan has been created, the execution time of this approach is short. However, the approach suffers from several disadvantages. First, a design plan must be created for each circuit to be designed. This is a difficult task and requires the knowledge of a skilled designer. It was reported in [14] that the average time to create such a plan was four times longer than manually designed circuit.

In addition to that, the design plan is technology dependent. This means, when the process technology migrates to a new technology, a new design plan must be updated which again requires analogue experts intervention.

Another limitation to the approach is the accuracy which is generally limited. In order to derive the design equations for the design plan, they are bound to be simple. This will result in large deviations in the performance metrics when modern process technologies are used.

This section reviews some of the tools that were developed using this approach.

IDAC

IDAC [15] is one of the first and well-known approaches for knowledge-based design techniques. It was developed in 1980 and support quite large design variety of circuits such as amplifiers, comparators and A/D converters. This tool relies upon a library of circuit design plans. Each design plan contains a set of design equations for particular topologies created by an experienced designer.

From a set of design specifications, a design plan for a particular circuit topology is executed. From this execution, a set of design parameters will be known and a circuit simulator is used to verify the performance of the circuit. If it fails to meet the specifications, the parameters are adjusted and the design plan is executed again.

IDAC contains a predefined library of circuit designs, so the design time is short for circuits already in the library. However, if the designer wants to make changes to the topology for example to improve the performance, a completely new set of design plans must be developed.

While the execution time might be fast for a circuit already in the library, IDAC presents several disadvantages. As mentioned above, the design plan is created by expert designers thus it is highly dependent on the experts whenever a new design or topology needs to be developed. Second, it is not possible to solve equations for high accuracy device models, thus the method is limited to simple models. This yields relatively poor estimation of the circuit performance.

OASYS

OASYS [16] was developed in 1989 at Carnegie Mellon University. This method describes the design problem in a hierarchical style implementation where the circuit is partitioned into several sub-blocks. From the design specification, the tool selects a suitable topology. This topology is then divided into several sub-blocks that correspond to the performance specification. In this way, the problem is decomposed into separate design tasks. There is a possibility that there may be several sub-blocks with the same functionality. The tool generates a range of possible options, or “styles” and selects the one with the best performance. This is called style selection.

A translation process in the methodology will map the performance specification to the sub-block. In a design, there might be several hierarchical levels and style selection steps and translations. At the bottom level (transistor level), simple device models are used to determine the device sizes based on a knowledge-based approach. Sometimes, there might be a discrepancy in the estimation of the performance of low level blocks. This is overcome by utilizing backtracking strategy to refine the design. This is an iterative process and may be seen as simple form of optimization.

The method forms some sort of reusability since the sub-blocks can be used repeatedly in a large range of circuits. The disadvantages of this method are first, the use of simple device models to determine the device size which yields relatively poor estimation of the performance. Second, the task of creating design plan consume a lot of design time as reported in [16] where the creation of the first design plan required 18 months to be completed.

BLADES

BLADES (Bell Laboratories Analogue Design Expert system) [17] relies on artificial intelligence to partition and size the circuits. As with OASYS, the strategy is to divide the circuit into several sub-blocks. For example, an operational amplifier might consist of a differential input stage, gain stage and output stage. The rules on how to divide the circuit into sub-blocks are written in “if-then” statements. For the operational amplifier, the tool consists of about 250 different rules.

The bottom level is the transistor level. The transistors are sized in a similar manner to the sub-block composition where a set of rules is used to size the transistor. Here, the decision about the size is decided based on the rules given in the combination of look-up tables where the simulated results for each sub-block are stored.

As with other knowledge-based approaches, the disadvantage of this tool is the requirement to create the design rules for each adjustment and/or addition to the topology of the circuit.

2.2.2 Optimization-Based Circuit Sizing

Knowledge-based techniques rely on design plans created for specific topologies. In other words, it is a topology dependent approach. In order to increase the generality of circuit sizing and make it independent of circuit topology, optimization-based design was developed. In this approach, the decision to size the circuit is based on an optimization algorithm rather than design plan. Two important stages of this approach are optimization and evaluation as depicted in figure 2.2.

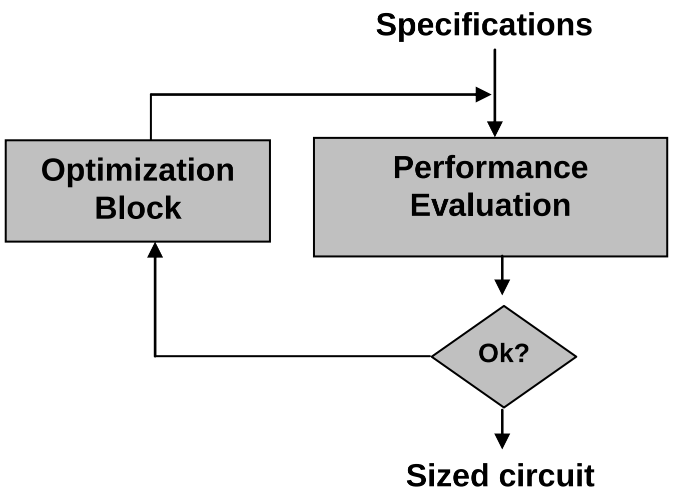


Figure 2.2 : optimization-based design

There are two types of optimization-based design. The first type is based on a circuit simulator such as SPICE which is used to evaluate the performance of the circuit. A circuit simulator is called at each iteration to determine the performance for a set of design parameters. This approach is called simulation-based optimization.

Another type that is used is equation-based optimization. In this approach, a set of equations that relate the circuit performance and the design parameters is derived. These equations are used to evaluate and determine the performance for a set of design parameters. This process is continued iteratively until the performance is optimized.

2.2.2.1 Equation-based Optimization

Equation-based optimization uses equations to evaluate the circuit performance as oppose to the use of circuit simulator for the simulation-based optimisation. The equations can be derived manually or using symbolic analyzers [18, 19, 20].

The advantage of equation-based optimization is in the execution time since the performance evaluation is performed by evaluating symbolic equations directly [21]. Thus, the equation-based approach offers significantly shorter execution times compared to simulation-based optimization.

The accuracy of the performance predictions is extremely reliant on the design equations. Manually derived equations are usually simplified compared to equations derived by symbolic analyzers. Most of the equations are based on simple device models and are therefore not accurate enough to be used in modern process technologies. Sometimes, if high accuracy device models are used, the equations created are based on approximations in order to reduce the size of expressions for the performance metrics. Small expression sizes will increase the computational efficiency in the expense of accuracy. This is one of the disadvantages in this approach, in that there is clearly a trade-off between accuracy and speed.

Furthermore, using symbolic analyzers to generate the equations automatically will increase the setup time for this approach. With designer instruction, a symbolic analyzer will generate the equation expression for each performance metric. Thus, introducing new types of performance metric into the symbolic analyzer can be time-consuming.

Another disadvantage of this approach is that the generality of the method is limited by the ability to derive the equations for the performance. A symbolic analyzer can be used to derive small-signal performance metrics but for other performance (for example one that uses time-domain analysis such as slew-rate), there is no method to automatically generate the equations. These type of equations need to be derived manually. For a different device model, new equations must be derived to include the additional parameters of the device model. On top of that, the equations are created by an experienced designer and stored in a library. Thus, the method is often only applicable to a predefined topology in the library.

This section reviews some of the tools that have been developed using this approach.

OPASYN

OPASYN [22] was developed in 1990 at the University of California in Berkeley, USA, and uses simple analytical equations to synthesise and optimize a circuit. It features a design database that contains information on each step in the design flow, including heuristic selection of circuit topology, circuit sizing and optimization and circuit layout.

From a set of performance specifications, a circuit topology is selected from the database. The selection is done using a decision tree where all available topologies are classified according to some key criteria and analytical models is used to size and optimize the circuit. The models consist of manually derived symbolic design equations, netlist descriptions of a particular topology, independent design parameters and upper and lower bounds for the design parameters. The optimization method used is a steepest descent algorithm and to avoid local-minima problem, the optimization is carried out on several starting points.

The disadvantage of the tool is the accuracy of the models. It was reported in [23] that the models have an error of over 200% when compared to SPICE simulations. Although fitting parameters are added to improve the model, the error is still in the order of 20%.

Maulik

Maulik [24] was developed at Carnegie Mellon University in Pittsburg, USA. This tool selects the topology and size the circuit simultaneously. Additional optimization parameters are used to determine the topology such as the type of the input stage (for example cascade or not). The performance functions are computed from circuit equations and these are used to size the circuit.

Maulik uses a relaxed DC formulation to solve for the correct DC operating point. In this approach, Maulik uses Kirchhoff's law for the DC operating point equation and this is made as a part of the cost function. With a relaxed DC formulation, Maulik avoids the need to re-evaluate the DC operating point at every iteration.

One of the disadvantages of this tool is the requirement to derive the equations manually which leads to the simplified expression thus limiting the accuracy.

GPCAD

GPCAD [25] is a device sizing tool dedicated to the design of operational amplifiers. It uses geometrical programming (GP) to formulate the sizing task. This is done by writing the design equations (i.e. the cost function and inequality constraints) as posynomial equations. This results in a convex optimization from which a global optimum point can be found in a relatively short time.

Even though the geometric programming formulation simplifies the optimization task and reduces the optimization time, this method suffers from an accuracy problem due to the limitation of using high accuracy models that cannot be formulated as posynomials easily [23]. Furthermore, this tool does not include automatic generation of the equations thus limiting the usage to only predefined circuit structures.

2.2.2.2 Simulation-Based Optimization

Simulation-based design uses a standard circuit simulator in the optimization loop to evaluate the circuit performance. In this way, the method can handle a large variety of analogue circuits.

One of the advantages of this approach is that the predicted performance will have the same accuracy as the models used in the circuit simulator, i.e., the same accuracy as obtained by manual design. Even with the new process technologies, the level of accuracy can be maintained if the process model is used in the simulation.

Another advantage of simulation-based design is short setup time. This is true as long as the circuit performance can be measured using the output of the circuit simulator. The only requirement is to create the test bench in the simulator environment. The test bench describes the simulation environment to measure each performance function for the optimisation.

Furthermore, the generality of simulation-based design is high since the performance can be defined just by extending the test bench. Thus, new circuits can be included easily as long as the circuit simulator can be used to extract the performance metric.

The only disadvantage of this approach is the execution time. In the simulation-based approach, a circuit simulator is called at each of the optimization run. Some of the performance functions such as slew rate which require time domain simulation may consume significant amounts of simulation time. However this factor can be mitigated with the continual advance of computer hardware.

This section reviews some of the tools that have been developed using this approach.

DELIGHT.SPICE

DELIGHT.SPICE [26], was developed in 1980's at the University of California, Berkeley, USA. The tools combined an interative optimisation based design called DELIGHT with a standard circuit analysis program, SPICE.

The tool also derives the sensitivity of the design parameter variations to the performance functions which enable design centring and yield optimization. The optimization algorithm in DELIGHT.SPICE uses a subset of worst performance and constraint functions to direct the searching process.

The algorithm consists of 3 phases: phase I, the optimisation algorithm tries to decrease the hard constraint violation. Hard constraint is the constraint that must be satisfied and do not take part in design trade-off. In phase II, the worst normalised values of the objective functions and soft constraint are improved while maintaining the hard constraint satisfaction. In phase III, the worst normalised value of objective functions is improved while both the hard constraint and soft constraint are satisfied.

However, the tool still requires several hours to perform the optimization and in addition to that, a good starting point is needed for the optimization process in order to avoid divergence problem in SPICE [27].

FRIDGE

FRIDGE [28] is a simulation-based optimization approach that performs global searching techniques together with a gradient search for the optimization algorithm. The tool uses modified simulated annealing for the optimization. Instead of slowly cooling scheme of traditional simulated annealing method, this tool uses adaptive cooling where a series of fast cooling and reheating method are used.

The optimization is divided into two stages. The first is to quantize the design parameters according to a grid and the performance of the design parameters that corresponding to one node of the grid is stored. This is used to avoid repeated simulation of the same node. Once the global optimization is completed, a gradient based optimization is used to search in the vicinity of the best grid point.

ASTRX/OBLX

ASTRX/OBLX [4] have been developed in 1996 at the Carnegie Mellon University. The tool relies on asymptotic waveform evaluation (AWE) [29], encapsulated device evaluators, simulated annealing and relaxed DC formulation to size and optimize the circuit.

AWE [29] is used to reduce the long simulation times normally associated with circuit simulators in simulation-based design and low accuracy that is normally achieved in simple models used in equation-based design. AWE uses a reduced complexity model to predict the small signal circuit performance. This approach is efficient to analyse linear circuits is considerably faster than using a SPICE-like simulator. The rest of the performances (other than small signal) are computed from circuit equations.

Simulated annealing is used to solve the optimization problem. The constrained optimization formulation given in equation 2.1 is solved in an unconstrained fashion. Here, x is the independent variable – size of semiconductor devices or passive components value that need to be find, $f(x)$ is a set of objective functions that need to be optimized, $g(x)$ is a set of constraint functions that specify the specifications and w_i is the scalar weight to balance the competing objectives.

$$\text{Minimize } \sum w_i f_i(x) \text{ , } \; g(x) \leq 0 \qquad (2.1)$$

The constrained optimization formulation is converted to an unconstrained optimization with the use of additional scalar weights for the constraint parameters. As a results, the goals become a minimization of scalar cost function $C(x)$, defined in equation 2.2.

$$C(x) = \sum w_i f_i(x) + \sum w_j g_j(x) \qquad (2.2)$$

To solve the DC operating point for each perturbation of design variables, a relaxed DC formulation was used in this tool. Kirchhoff's Law was used to solve the DC operating point and this is included in the constraint function of the optimization formulation similar to Maulik [24] method.

One drawback of this tool is the inability of AWE approach to model nonlinear circuit behaviour. Furthermore, the approximation of the circuit transfer function with a low-order model limits the accuracy of the method.

ANACONDA and MAELSTROM

Both of these simulation-based techniques were developed at the Carnegie Mellon University in 1999 for MAELSTROM [30] and 2000 for ANACONDA [31]. The difference between these two is in the optimization algorithm. MAELSTROM uses a combined genetic and annealing algorithm whereas ANACONDA uses a stochastic pattern search.

The tools rely on three key concepts: simulator integration, global search techniques and a parallelism approach to reduce the overall computation time where the searching tasks and circuit evaluations were distributed across a network of cluster workstations.

The optimization formulation was adopted from the OBLX strategy where a constrained optimization formulation that is solved in an unconstrained fashion was used. As with OBLX, this technique introduce scalar weight values to the optimization formulation and the goal becomes minimization of a scalar cost function.

The optimization engine in MAELSTROM is based on a combination of simulated annealing and genetic algorithm. The simulated annealing engine is called Anneal++ that offers a range of annealing cooling schedules, move selection techniques and dynamic update of the cost function weights. The genetic algorithm is used for the purpose of parallel search. The combination of genetic algorithm and annealing in this method is known as the Parallel Recombinative Simulated Annealing (PRSA) as proposed by Goldberg [32].

ANACONDA uses a combination of population search of circuits with pattern search in finding the circuit solution. The pattern search method proposed by Torczon [33] is a direct-search techniques that sample cost function in a deterministic locus around a given solution point and use this sample to construct a deterministic direction and distance to a probable better solution. The combination of population search and pattern search helps the optimization engine to explore a diverse set of samples of the objective (cost) surface.

2.4 Optimization Techniques

One of the key components in an optimization-based approach is obviously the optimization block. The function of this block is to optimize the design by searching for the best solution points which are determined by the design parameters. In this context, the purpose of the optimizer is to find the design parameters that will produce the best performance value. The process between the optimizer and the performance evaluator is an iterative one where the performance for a particular design parameters will be evaluated and the design parameters will be changed from run to run in order to improve the performance. The process will be continued until the optimization objective or stopping criteria has been met.

Generally, with the rapid development in optimisation algorithms, the algorithms can be divided into two main categories: population based and single initial solution based. The difference between the two is the type of initial solution. Population based approach starts with a set of solutions called a population while single initial solution starts with one initial solution. Recently, an optimisation approach that uses a heuristic process consisting of many optimisation runs starting from different initial points has been proposed [34]. In this way, the optimisation process becomes a group of individual optimisation runs. The rest of this section will review some of the optimisation techniques that have been used for the circuit optimisation.

2.4.1 Direct search Optimisation

This section will discuss several optimisation methods known generally as direct search algorithms. Box *et al* [35] identified three main types of direct search algorithms: tabulation, sequential and linear methods.

2.4.1.1 Tabulation Method

In this method, a user chooses number of points either using a random tabulation or a grid tabulation strategy. The objective function is evaluated at each point and the point with the lowest function value is returned as the optimum solution.

2.4.1.2 Sequential Method

In this method, a geometrical figure of the same dimension as the decision space (design variables) are created and the performance function is evaluated at each of the geometrical nodes (vertices) in order to find the minimum point. Decisions are taken on the basis of comparing function values corresponding to the vertices of the geometrical figure. Evolutionary operation [36] was the first sequential method developed. This is followed by an improved algorithms known as simplex method [37]. The geometrical figure used in the simplex method has n + 1 nodes where, n represents the number of design variables. Thus, the simplex is a triangle for n=2, tetrahedron for n = 3 and hypertriangle for n > 3. Once the figure has been determined, the performance is evaluated at each of the nodes and a convergence test is applied. The convergence is said to be met if the standard deviation of the function values at all vertices are less than a user-defined level (to be determine by trial and error).

2.4.1.3 Linear Method

This method involves a set of searching sequences along lines in the decision space and can be divided into two main categories : univariate search (and its derivatives) and Powell’s method [38] (and its derivatives). In univariate search, the optimisation starts with user specified initial values of the n design variables. Each of the design variables will be evaluated one at a time to determine the performance function and the design variable will be adjusted until the performance function is minimised. The optimisation is stopped when a user-defined maximum iteration count is exceeded or the performance function at any point falls below a user-defined acceptance level. Even though the univariate search is simple to implement, it has two major limitations. Firstly, the search is carried out sequentially and secondly, the search procedure is completely deterministic which would generally result in a premature convergence to some relatively poor local minima [39]. In addition to that, the convergence rate is relatively slow as the minimum point is approached. The slow convergence rate is enhanced by introducing a pattern move algorithm [40] that involves two procedures: the exploratory move and pattern move. In the exploratory move, a fixed user-defined increment is applied to the initial points. The performance function is evaluated at this new points with the increment. If the performance is minimised, the incremented point will be the new base point. Following a successful exploratory move, a pattern move procedure is performed where both the previous base point and the new base point are connected and used as the new searching direction. Even though the pattern move algorithm improves the convergence rate, there are several other methods that have been developed to improve the efficiency of the algorithm. Bandler introduced `Razor Search' [41], in which a second increment size is added if the initial increment manage to minimise the performance function. This new increment size is related to the distance between the previous two base points. A second-order pattern move was proposed by Massara and Fidler [42] that involves the use of original pattern move followed by a searching along a quadratic curve fitted to the last three base points. Emery proposed the `spider search [43]' which performs the exploratory move in a randomly selected sets of orthogonal directions.

2.4.2 Gradient-search Optimisation

Gradient methods involve the use of first and/or higher derivatives of the objective function to determine a suitable search direction. There are three main categories in this method: steepest descent (the use of first order derivatives), Newton's method (second-order derivatives) and quasi-Newton methods.

2.4.2.1 Steepest Descent Method

The steepest descent method (SDM) [44] is a gradient search method where it uses the derivatives to find the downhill direction of the objective function. To find a local minimum to an objective function, from a starting point, a search is conducted for a minimum points towards the negative gradient of the function. This method was used in one of the earliest reported applications of optimisation to electronic circuit design for the design of lossy ladder filters [45]. The method of steepest descent is defined by the iterative algorithm based on equation 2.3.

$$x_{k+1} = x_k - \alpha_k g_k \tag{2.3}$$

Where α_k is the scalar for function minimization. In this equation, the starting point of the minimization is x_k and from this starting point, a search is conducted along the direction of the negative gradient $-g_k$ to find the minimum point on this line. The minimum point is denoted by x_{k+1} .

The steepest descent method or gradient method has several disadvantages searching for optimal solutions. Firstly, the convergent speed of the method is slow due to the step size in the searching process in a single line search. Furthermore, the derivation of a system function is difficult and prone to approximation errors [46]. Also, the solution may not be the global optimum solution for the problem. The reason for this is that the method will only converge to a local minimum based on the starting point. Hence, for a poor initial starting point, the resulting solution may be far from the global minimum.

2.4.2.2 Newton's Method

This is one of the most widely used optimisation method based on gradient calculation [47]. In this method, from an initial guess x^o, a correction vector, Δx is determined to find the minimum point, x^{min} of a quadratic function. From a Taylor series expansion and differentiation, an expression for x^{min} as given in equation 2.4 is obtained where g is the first partial derivative and H is the

$$x^{\min} = x^o - H(x^o)^{-1} g(x^o) \qquad (2.4)$$

Hessian matrix of the second partial derivatives. From this expression, a new point x^{r+1} is derived and determined according to a user-defined line search strategy.

The Quasi-Newton method is based on Newton's method but without the explicit evaluation of the Hessian matrix and its inversion, which may cause divergence. The quasi-Newton methods use an approximation to the Hessian inverse [48]. Thus the Hessian inverse, H^{-1} is replaced by H^r representing the approximation after r iterations.

2.4.3 Simulated Annealing

The simulated annealing optimisation method was investigated by KirkPatrick *et. al* [49] in 1983. It uses the mathematical analogy of heating and controlled cooling processes to solve for an optimal solution. The proposed method is based on a procedure to make the strongest possible glass. The procedure starts with heating the glass to a high temperature so that the glass is liquid (atom move freely). Then, the temperature of the glass is slowly lowered so that the atom can move and relax into a stable condition. The slow cooling process is known as annealing.

The equation for the probability of a system to be at the energy level, E_0 is given by equation 2.5.

$$\rho(E_0) = \exp\left(\frac{-E_0 / k_B T)}{Z(T)}\right) \qquad (2.5)$$

Where k_B is the Boltzmann constant, T the temperature and $Z(T)$ is a normalizing function.

The standard simulated annealing (SA) procedure starts with generating an initial solution randomly. A new solution is generated by perturbation of the previous solution. The objective function value of the new solution is evaluated and compared with the previous solution. A move is made to the new solution if it has a better value than previous value or the probability function $\rho(E)$ is higher than a randomly generated number. Otherwise a new solution is generated and evaluated. Simulated annealing employ uphill moves to avoid local minima. Therefore, the method has a better capability to find a global optimum solution in a given problem.

2.4.4 Genetic Algorithm

The Genetic (or Evolutionary) Algorithm is one of the stochastic methods that is widely used in optimization. Stochastic methods incorporate probabilistic (random) elements in the algorithm. This approach is based on the mechanics of natural selection and natural genetics where they combine the fittest individuals among the population in order to search for the best individual [32].

The random nature of Genetic Algorithms may not find the absolute best solution, but it has a greater chance of finding a good solution, quickly, for difficult problems [50]. On top of that, Genetic Algorithms are a population based algorithm making it a suitable candidate to search for a several optimal solutions in one run.

The algorithm consists of several stages including coding the problem (chromosome representation), generating initial population, evaluating fitness function, crossover and mutation. It starts with a randomly generated population which will be evaluated and scored according to the performance. From this population, the next generation will be bred using selection and recombination procedure to produce new offspring. As with genetic of living organisms, combination of two good individuals often will produce offspring that are better adapted to the environment, thus having a better fitness score. A small mutation probability is then added to the new offspring. This is the stage that mimics the mutation that happens in living organisms. In nature, mutation happens when the genetic of the organism is accidentally changed that will change the DNA of the individual. In this algorithm this situation is carried out by selecting few genes in the chromosomes and randomly changing them to a new gene but the mutation occurs depending on the probability that has been defined. As in biological systems, the mutation adds new variation to the population. Once the new generation has been generated, the whole process will be repeated until the final number of iterataions or stopping criteria is met. Figure 2.3 shows a flowchart of the algorithm.

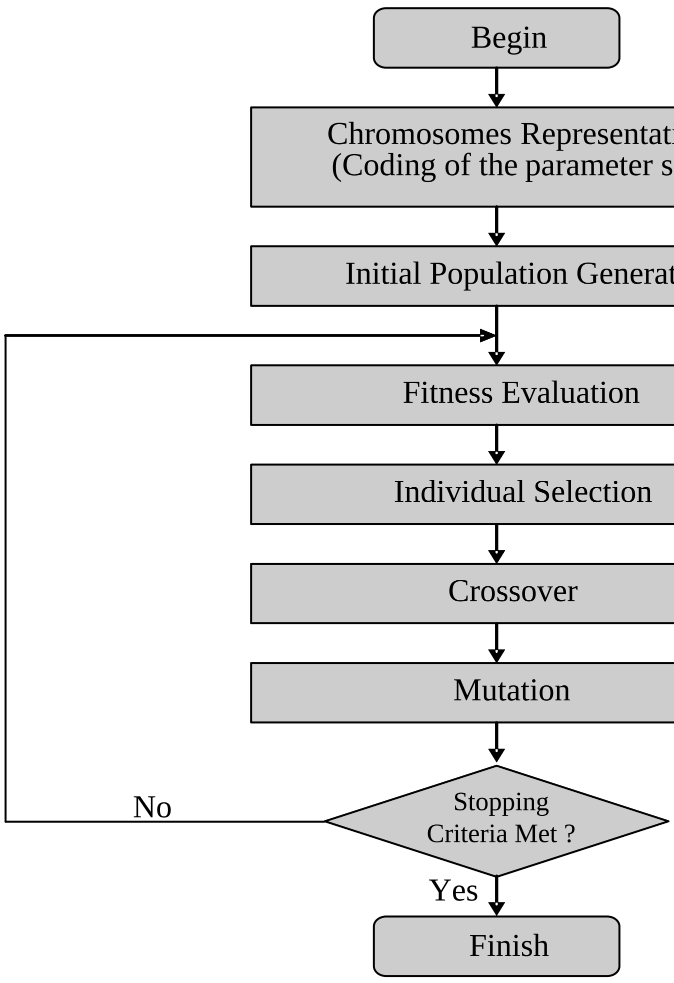


Figure 2.3: Flowchart of Genetic Algorithm

Prior to the optimisation, several parameters of the genetic algorithm need to be specified. The parameter analysis is beyond the scope of this research as the objective of the research is to demonstrate the methodology that can be used to optimize the performance and yield of a system level design and the GA is a tool used for the optimization. Therefore, the parameter settings for the genetic algorithm presented in the thesis were chosen based on the DeJong [108] recommendation. However, in certain circuit examples, some of the parameters such as the population size might be different from the recommended setting in order to reduce the optimisation time. The GA control parameters used in this thesis are shown in table 2-1. Figure 2.4 shows an example of an output report from a multi objective optimisation showing all the control parameters used by the GA.

GA Parameter	Setting
Population size	50
No. of generation	100
Crossover type	Single point
Crossover probability	0.6
Mutation probability	0.01

Table 2-1: GA parameter setting

```
GA PARAMETERS
-------------------------------------------------------
Population Size ->50
No. of generations ->100
No. of Functions ->3
No. of Constraints ->0
No. of real-coded variables ->11
Selection Strategy is Tournament Selection
Variable bounds are rigid
Cross-over Probability ->0.600000
Mutation Probability for real-coded vectors -> 0.010000
Results in a file
```

Figure 2.4: Example of genetic algorithm report

The population size parameter is the initial random number of individuals created for the optimisation. A large population will consume higher optimisation time whereas a small population can lead to a premature convergence which will reduce the ability to find the best solution. Even though a population size of 50 was used in most of the examples shown in this thesis but for a complex circuit such as PLL in chapter 7, smaller population size is used in order to reduce the optimisation time. The number of generations represents the number of iterations needed before the optimisation is terminated. However, convergence criteria can be added to the algorithm that can be used to stop the optimisation early if the criteria are met. An example of using a stopping criteria is shown in chapter 4. Crossover is a process of creating ‘offspring’ from two individuals by swapping part of their chromosomes (GA string). This process is intended to simulate the process of recombination that occurs to the choromosomes during sexual reproduction in biology system. One of the common forms of crossover used in this research is single point crossover where a single point of exchange called crossover point is set at a random location in the two individual genomes. One individual will contribute all the parameters from before that point and the other will contribute parameters from after that point to produce an offspring. However, the crossover does not always occur and this is based on a determined crossover probability. The probability of crossover occurring in this research is set at 0.6 or 60%. When there is no crossover, the parents are copied directly to the new population. Another GA control parameter is called mutation. This process is used in order to make sure the individuals are not all exactly the same by changing one parameter from the GA string. The rate of the mutation occurs is controlled by the mutation probability. All the examples in this thesis use 1% or 0.01 mutation probability.

2.4.5 Multi Objective Optimization

Circuit performance is a function of designable parameters. The design goal is to find a parameter set solution that meets all the performance functions and any imposed constraints. The optimization formulation for more than one objective function is called multi-objective optimization which can be generally stated as given in equation 2.6.

$$\begin{aligned} &\text{Minimise / Maximise } f_m(x), m = 1,2,....M \\ &\text{Subject to } g_j(x) \geq 0, \quad j = 1,2,...J \end{aligned} \qquad (2.6\ )$$

Where $f_m(x)$ is the set of M performance functions and $g_j(x)$ is the set of J constraints. In a design that involves multiple conflicting objectives there is not usually a single optimum solution which simultaneously optimizes all objectives. The outcome from multi-objective optimization is therefore a set of optimal solutions.

The outcome of the multi-objective optimisation is a set of solutions that define the objective space with the number of dimensions equal to the number of objectives. Figure 2.5 shows the relationship between the parameter space and objective space. Each point in the parameter space is a solution that corresponds to a point in the objective space. The black curve on the objective space is called the Pareto front and all solution points lying on this curve are called Pareto-optimal solutions. Point B in the solution space is an example of a non-Pareto optimal point since a more optimal solution exists, point (A). Several algorithms [51] for Multi-Objective Optimisation have been proposed and will be discussed in the following sub-sections.

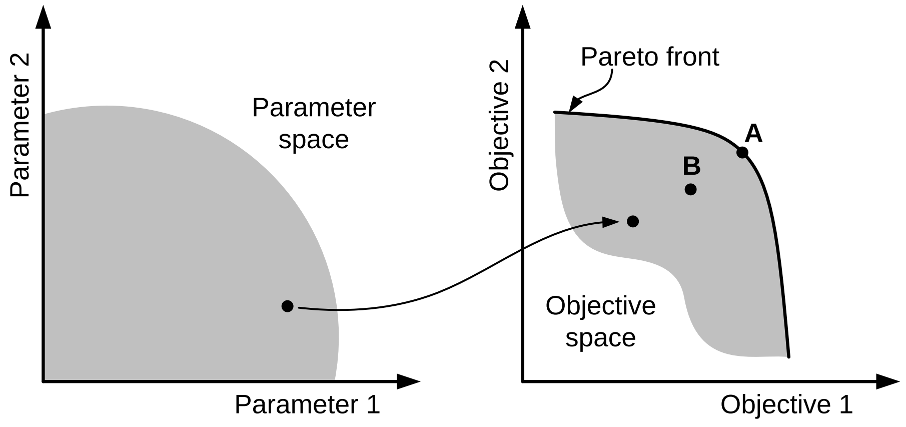


Figure 2.5: Relationship between parameter space and objective space

2.4.5.1 Weighted-based Genetic Algorithm

One of the simple algorithms used for multi-objective optimisation is Weighted-Based Genetic Algorithm (WBGA) [51, 52]. In WBGA, all the performance measures are combined into a single objective using a weighted summation method as shown in equation 2.7. W_m is the weighting for each of the performance functions, f_m.

$$\sum w_m f_m(x), m = 1,2,...M \tag{2.7}$$

In WBGA, the weight of the summation is determined by Genetic Algorithm. This is done to overcome the problem of finding suitable weight parameters that normally associated with classical weight summation method.

2.4.5.2 Non-dominated Sorting Genetic Algorithm-II (NSGA-II)

NSGA-II [51] is one of the widely used evolutionary algorithms for multi-objective optimisation. This algorithm is categorized as elitist-based as it allows the elite individuals to be carried over to the next generation in order to ensure that the population’s best solution does not deteriorate. In this way, a good solution found early on in the run will never be lost unless a better solution is discovered.

The algorithm starts by creating an offspring population Q_t from a parent population, P_t. These two populations are combined together to form R_t (the combination of P_t and Q_t)of size 2N (N is the size of each population). Then, a non-dominated sorting approach is used to classify the entire population R_t. This step checks for non-dominated points among the individuals and sorts accordingly. The next step is to generate a new population with size N and fill this population with solutions of different non-dominated fronts from the previous sorting. The filling starts with the best non-dominated front, followed by second best and so on. Since the population size is N which is smaller than the size of R_t which is 2N, not all fronts can be accommodated in the new population. All fronts that cannot be accommodated in the new population are discarded. Sometimes, there exists a condition where the last front has more solutions (individuals) than the available space in the population. In this case, a crowding distance metric is used to choose which members of the last front are placed in the new population. Figure 2.6 illustrates the strategy employed by NSGA-II. Once the new population is filled with all fronts, the selection, crossover and mutation operators will be applied to this population to create new offspring and the whole process is repeated again until the final number of generations has been reached. The step-by-step algorithm flow in NSGA-II is outlined in figure 2-7.

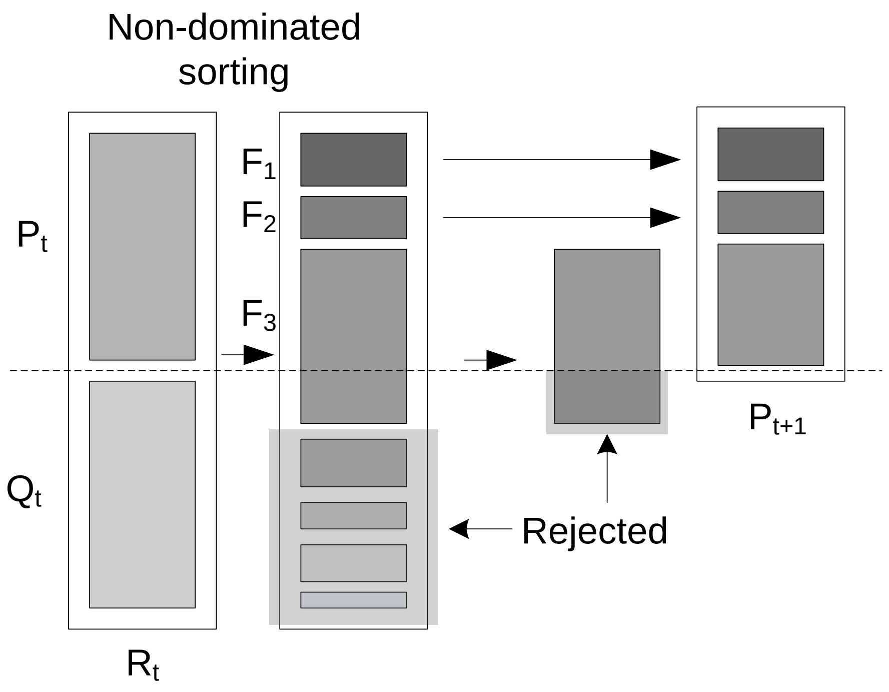


Figure 2-6: NSGA-II Procedure

```
NSGA Algorithm

- Generate initial random population, size N.
- Create offspring population.
- Combine parent and offspring population to form $R_t$. ($R_t=P_t \cup Q_t$)
- Perform non-dominated sorting and identify fronts, $F_i$ (i=1,2...etc)
- Set new population, $P_{t+1} = 0$, and fill $P_{t+1}$ with $F_i$ ,($P_{t+1} \cup F_i$) as long as $|P_{t+1}|+|F_i|<N$.
- Perform crowding sort and place most widely spread solution in $P_{t+1}$
- Create offspring populaiton $Q_{t+1}$ from $P_{t+1}$ and repeat until last number of generation.
```

Figure 2-7: NSGA-II algorithm

Other than WBGA and NSGA-II algorithms, there are several other evolutionary algorithm for multi-objective optimisation such as NPGA (niched Pareto genetic algorithm) [53] and SPEA (strength Pareto evolutionary algorithm) [54]. NPGA is based on a non-domination concept as NSGA-II and uses binary tournament selection for the selection procedure. The motivation behind the procedure is coming from the genetic algorithm (GA) theoretical studies [55] that show the advantage of tournament selection in terms of better growth and convergence properties. SPEA was proposed by Zitzler and Thiele [54] and is one of the elitist-based algorithm similar with NSGA-II. The elitism is introduced by explicitly maintaining an external population. This population contains a fixed number of the non-dominated solutions that are found in the beginning of a simulation. At every generation, newly found non-dominated solutions are compared with the existing external population and the resulting non-dominated solutions are preserved.

2.5 Statistical fluctuations in integrated circuit

During the fabrication process of integrated circuit, the components and their interconnections are fabricated simultaneously in a series of process steps. Statistical variations in these processing steps lead to variations in the component parameters and hence in circuit performances. If the performance of the integrated circuits is measured, the results will be found to have deviated from the nominal (designed) values. The extent of this deviation may be such that the performances of the circuit fail to meet the specifications. This will result the manufacturing yield to be less than 100%.

The manufacturing of integrated circuits often suffers from statistical fluctuations (variations) in the fabrication process. The variations can be divided into two types: inter-die (die-to-die) and intra-die (within-die) variations. As described in chapter 1, these fluctuations are getting worse in deep submicron process technology. It was reported that the magnitude of intra-die channel length variations has been estimated to increase from 35% of total variation in 130nm, to 60% in 70nm process [56]. Statistical variations can cause a failure in the manufactured circuit. These failures can be either catastrophic or parametric. Catastrophic failures cause a change or unexpected functionality to the circuit while parametric failures cause the performance of a circuit to deviate from the targeted value. The ratio of circuits that meets the specifications to the total number of fabricated circuits is called the yield. A low product yield implies a financial loss to the IC manufacturer and due to the high correlation between high yield and high profits, the yield has been a big concern. The design approach to maximize the yield during the design stage is known as Design for Yield or Design for Manufacturability (DFY/DFM).

2.6 Parametric Yield Maximisation

Yield maximisation techniques attempt to find a suitable set of nominal design parameters such that most of the circuit that are manufactured will meet the specifications of the performance functions. The performance space of a design is defined as a series of performance of interest by n_φ as given in equation 2.8.

$$\varphi = (\varphi_1, ...\varphi_{n_\varphi}) \qquad (2.8)$$

Parameter space is defined by the set of design parameters that determine the performances, by the n_p vector as given in equation 2.9.

$$p = (p_1, ..., p_{n_p}) \qquad (2.9)$$

A manufactured circuit will be considered acceptable if all of its performances fall within acceptable limits (meet the specifications) which can be represented by equation 2.10.

$$\varphi_k^L \le \varphi_k \le \varphi_k^U, \quad k = 1,...,n_\varphi \qquad (2.10)$$

Where, φ_k^L is the low limit and φ_k^U is the upper limit. Equation 2.10 defines a region of acceptability, A_φ in the n_φ dimensional performance space. The specifications determine a region in the performance space where the circuit is acceptable. Figure 2.8 illustrate the acceptability region for a 2 dimensional performance space.

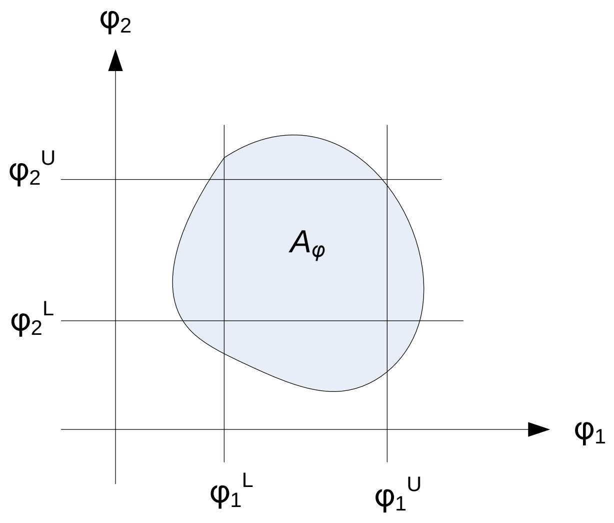


Figure 2.8: Acceptability region in performance space

The circuit parameters, p can be modelled as functions of their deterministic nominal values, p^0 and a set of random variables that characterize process variations, ξ, as given in equation 2.11.

$$p = p(p^0, \xi) \qquad (2.11)$$

The circuit performances can be modelled as a functions of the nominal parameter values and the statistical variations shown in equation 2.12.

$$\varphi = \varphi(p^0, \xi) \qquad (2.12)$$

The region of acceptability in the variations space, $A_\xi(p^0)$ consists of all the possible combinations of variations that can occur in the manufacturing of a circuit which specific nominal parameter values do not result in acceptable performance. The region of acceptability can be defined by equation 2.13.

$$A_\xi(p^0) = \left\{ \xi \mid (\varphi^L \le \varphi(p^0, \xi) \le \varphi^U) \right\} \qquad (2.13)$$

The yield of a design can be calculated in the design parameter space or circuit performance space. In performance space, yield is formulated as given in equation 2.14.

$$Y = prob\left\{ \varphi \in A_\varphi \right\} = \int_{A_\varphi} f_\varphi(\varphi) d\varphi \qquad (2.14)$$

Where $f_\varphi(\varphi)$ is the joint probability density function (jpdf) of the circuit performance φ. In the parameter space, yield is defined by equation 2.15.

$$Y = prob\left\{ p \in A_p \right\} = \int_{A_p} f_p(p, p^0) dp \qquad (2.15)$$

However, the calculation of yield is complicated by the fact that in either space, one of the two elements is not known explicitly: the statistical variations are known in the device parameter space but not in the circuit performance space, whereas the acceptability region is known in the performance space but not in the parameter space [57]. This makes yield prediction and maximization a difficult task and both spaces have to be considered. There are two important aspects related to yield prediction analysis and maximisation: variation analysis (the impact of variation towards circuit performance) and variation-aware design (the method to maximise the yield of a circuit design). In order to maximise the yield of a circuit design, the variation of the design parameters and the impact it has over the circuit performances must be analysed.

2.7 Variation Analysis

According to [58], analysis on the impact of variation to the circuit performance can be grouped into two main categories: worst case and non-worst case. In the first category, the analysis is done towards finding the circuit with the worst response with respect to the nominal value. The second category can be further divided into sampling and non-sampling methods. Method of moments, is one of the non-sampling methods which is based on the transformation of parameter tolerances into response tolerances. The objective of the transformation is to predict the distributions of the performance metrics based on the parameter distributions. The second category of non-worst case analysis, the sampling methods, are performance exploration techniques which perform circuit analysis at sample points in parameter space. The sample points may be chosen in a systematic (deterministic) manner as in simplicial approximation methods [61] and non-linear programming method [63], or randomly (statistically) as in Monte Carlo method.

2.7.1 Worst Case Analysis

The basis of this analysis is to identify the extreme (worst) values of performance resulting from the variations in parameter value. Since the only interested indication is the worst performance values, this technique does not requires the knowledge of the probability density function (statistical distribution) of the parameter values or the performance values. The procedure involves analysis of the worst case corners of the circuit performance based on some worst case combinations of the device parameters (e.g. slow-slow, fast-fast). The main drawback of this approach is that of identifying which combinations of the device parameters result in worst case corners [59]. Another limitation of worst case analysis is large overestimations of the circuit performance which is not suitable to predict the true relationship between the device parameter and their performance. [60] have systematically tackled the problems of worst case analysis for integrated circuits. They suggest to use worst case analysis in the intermediate stages of a design and only carry out worst case analysis of generic cell types and extrapolate to that of a larger proportion of the integrated circuit. On top of that, [60] suggest to treat process parameters as the basic component parameters. That is, starting from worst-case process parameters, worst-case device parameters are obtained using a process simulator. The worst case device parameters are then fed into a standard circuit simulator to obtain the corresponding performance values. With the process simulator, Nasif et al. [60] proved that the approach manages to avoid over-pessimistic results.

2.7.2 Simplicial Approximation

Simplicial approximation [61] is a method that approximates the boundary of the region of acceptability by deterministic sampling of the design parameters. In order to develop the boundary of the acceptability region, a sufficient points in the parameter space is determined. From initial design parameters, a circuit simulation is carried out to determine the satisfaction to the performance specifications. A search for the boundary is carried by varying one of the design parameters while maintaining all others fixed. At each step in the search, a circuit analysis is carried out to determine whether the circuit pass or not. The search is undertaken in all direction from the initial design parameters. The process can be repeated as many times as necessary to obtain the required approximation to the acceptability region (a region where all the parameters pass the specifications). Once the approximate region of acceptability has been obtained, a location of the tolerance region for the design parameters is determined. The tolerance region is obtained from the probability density function of the design parameters. With the tolerance region, it is easy to determine for each sample points, whether it lies within or without the approximation of the region of acceptability. The yield is estimated by dividing the number of sample points that lie in the acceptability region over the number of samples points generated. The main drawback of simplicial approximation is that it requires the acceptability region to be convex and simply connected. Unfortunately it is not possible to ascertain whether the acceptability region and performance specifications is convex or not. In addition to that, the computational cost of this approach is less only for a circuit with small parameter space (small circuit). With a bigger circuit (more parameters), the computational cost become high. This phenomena is termed as `curse of dimensionality

2.7.3 Monte Carlo Method

In the Monte Carlo approach for variation analysis, the sample points in parameter space are generated in a random manner to simulate the actual manufacturing process. The method directly mimics the process of random component value selection (including the correlations) by generating component values according to the known component probability density functions. The distribution of sample points in the parameter space can be either uniform or Gaussion (normal) function. The N circuit samples generated are then simulated using circuit simulator and their performance checked against the specification. Thus, a Monte Carlo analysis is akin to measurement made on N actual manufactured circuits. The yield for the circuit can be calculated as the fraction of samples that pass the specification, N_p over total number of samples, N. If N is sufficiently large, the yield provides a reasonable estimate of the yield that will be obtained from actual manufacturing process. As a rule of thumb, the number of samples is not fixed at the beginning. Instead, one or more performance-spread measures will be monitored and when no changes occur during the repeated simulations, the process can be terminated. One of the significant attributes of the Monte Carlo method is the accuracy of approach that is independent of the number of parameters. It is this property that allows Monte Carlo analysis to be employed for medium and large-size circuits.

2.8 Variation-aware Design

Variation-aware design deals with a design method to reduce the impact of process variations on the circuit performance. Generally, the approach for tolerance design can be divided into two phases [47]. First, the optimisation method (described in section 2.4) is used to find the nominal values of the parameters that will give the nominal optimum response. This phase is called the nominal-design phase. Several approaches have been developed for the nominal design that use analytical methods or simulation-based methods as described in chapter 2 in this thesis. In [62], the parameter distance that considers both the performance distance from the specifications and its sensitivity with respect to the design and operational parameters is used as the objective to find the optimum nominal design for the circuit. After the nominal design parameters has been solved, the tolerance of the parameters are determined from the response tolerance. Variation-aware design tries to minimise the impact of the parameter tolerances to the responses or performances. The method can be seen as an approach to maximise the yield of a circuit design which can be divided into two main categories : indirect and direct. The key difference between these two is the way the yield is considered in the design stage. The direct method considers yield as one of the objective function whereas indirect method does not.

2.8.1 Direct Method

Direct methods maximize the yield directly by employing yield as the objective function. Traditionally, this maximization is done at the end of the design process. In integrated circuits, yield can be expressed as multi-dimensional integral which can be evaluated numerically by Monte Carlo based methods. Monte Carlo simulation is the most straightforward statistical approach to predict the yield. In a Monte Carlo approach, the sample points in parameter space are generated in a pseudo-random manner to simulate the actual manufacturing process. For each sample, a SPICE simulation is performed and the resulting performance data sets are combined to derive the statistical distribution of the circuit performance as explained previously.

Monte Carlo analysis for a circuit design requires at least a circuit topology, device models and variations and mismatch model of the device parameters in the form of probability density functions (PDF). The process and mismatch model normally is given by the device vendor through their design kits. The set of values for the various device parameters are selected via a pseudo-random process from the known PDF. A circuit simulation is used to predict the performance of the circuit made up from the randomly selected set of parameter values. The procedure of random parameter selection and circuit simulation is repeated a number of times, and the parameter values and the corresponding predicted performance are recorded. The yield of the circuit would be found by comparing the predicted performance with the specifications, and establishing what fraction of the circuits satisfied the specifications.

One of the main advantages of Monte Carlo method is its dimensional independence characteristic [64]. What this means is that, the sample size required by random sampling is independent of the dimensionality(independence to the number of design parameters). For a comparison, the number of circuit simulations required for simplicial approximations is roughly exponential to the number of design parameters. This means in simplicial approximation, the number of circuits simulations is extremely large for a large circuit. This is not the case for Monte Carlo method.

In addition to that, the Monte Carlo method is very useful in hierarchical design for the purpose of exploring sub blocks performance variations. In a system level view, a design may be partitioned in to several sub blocks which can be realized by separate circuits. Generally, no specific performance requirements would have been placed on theses sub blocks, so the question of yield is not directly relevant to the sub blocks. One would have to estimate the performance spreads associated with the various sub blocks, perhaps with an initial allocating of allowed spreads among the properties of sub blocks and explore the trade-offs among them. In this case, Monte Carlo would be useful for its ability to provide estimates of the various performance distributions.

The disadvantage of the Monte Carlo method is the requirement to perform circuit simulations at every Monte Carlo point that would result to a very high computational cost. Several methods have been developed to reduce the computational cost. One of them is by using response surface method proposed by [65]. This two step method starts with parameter space sampling (as with the Monte Carlo method) with controlled simulations according to some design-of-experiments (DOE) scheme. For each performance characteristics, a response surface is then constructed by fitting a simple function of the device parameter to the simulated performance data. By initial screening, unimportant device parameters can be eliminated. In the second step, the evaluation of these simple response surface models analytically replaces full circuit simulation during the yield calculation. The limitation of this approach is the accuracy that is highly depend on the response surface models.

2.8.2 Indirect Method

The indirect method does not define yield as the objective function, hence the maximization towards yield is done indirectly by other alternative objective functions.

One example of indirect method for variation-aware design is design centering [66]. Several design centering algorithms based on statistical [67, 68] and deterministic methods [69, 70] have been proposed. This method attempts to place the nominal design in the centre of the acceptability region. Figure 2.9 shows how yield maximization is achieved by moving the parameter tolerance region towards the centre of the acceptability region. In this figure, P_1 and P_2 are the parameters, R_T is the region of tolerance of the parameters and R_A is the acceptability region of the design. By adjusting the nominal values of the parameters so that the region of tolerance can be moved towards the centre of the region of acceptability, the yield can be increased.

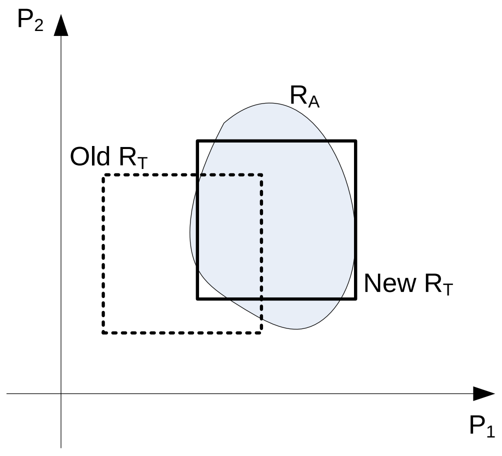


Figure 2.9: Design centring to maximize yield

Another design centering approach that indirectly optimizes the yield was proposed by [63]. Instead of geometric approximation, this approach explicitly approximates the acceptability region boundaries by the performance specifications. The author approximates the circuit performances based on quadratic function determined by an interpolation method. A nonlinear programming approach was used to optimize the performance function of a circuit with a minimum yield constraint.

2.9 Integrated Yield Optimization in Circuit Synthesis

Most of the earlier approaches in analogue circuit design consider yield as a separate step in the optimization process. In general, the synthesis starts with nominal-circuit design to meet the requirements and in the next step, the yield was evaluated and optimized by changing the nominal values. It is a great challenge to incorporate yield optimization as an integrated part of the circuit synthesis due to the large computational effort needed for such optimization. There have been several attempts with regards to the integration of yield in the optimization formulation. Some of these attempts will be discussed in this section.

2.9.1 ASTRX/OBLX Extension

The first attempt in this direction was proposed by Mukherjee [5]. In this approach, the author combines the statistical parametric variations, operating point variation and analogue circuit synthesis to form a system that can synthesize manufacturable analogue circuits. Mukherjee extended the synthesis strategy of ASTRX/OBLX to include operating range and parametric manufacturing variations to the methodology. The Non Linear constrained optimization Problem (NLP) formulation in ASTRX/OBLX is extended to a Non-Linear infinite programming (NLIP) formulation. The mathematical programming approach used is called infinite programming because of the infinite number of objective functions due to the inclusion of variation range in the objective functions. This approach employs worst case corners as the method to optimize the circuit design for performance and yield.

2.9.2 Simultaneous Yield and Robustness Optimization

In order to reduce the computational overhead of yield optimization, Debyser [6] proposed a technique that based on symbolic equations [71] and constraint satisfaction approach [72] to derive sizing plan and yield estimation plan for the optimization. Both plan (sizing and yield estimation) are simultaneously evaluated in the inner loop of a global optimization routine. The result of the optimization is a circuit design point that fulfills all the specifications and at the same time has pushed away the performances from specification boundaries under the influence of the yield.

The sizing plan of the analogue circuit is derived from a declarative analytical model. This model can be obtained through symbolic methods on the circuit’s graph topology. For the yield estimation plan, a reduced set of independent technology parameters is derived from a statistical transistor model. Then, a nominal design point and the variance of all performance parameters with respect to the reduced set of technology parameters are calculated. Using the perforamance variances, a yield representation based on two capability indices, C_p and C_{pk} is developed. Both of the indices strongly depend on the variance of the performances. Both of the sizing plan and yield estimation plan are used in the inner loop of the optimization routine to search for the best solution for performances and yield.

2.10 Summary

Due to the increasing demand for the design cycle time reduction for analogue circuit design, it has attracted huge interest among research community towards analogue circuit design automation. This chapter reviews some of the research works that have been devoted to the development of automation tools for analogue circuit. The automation tool development can be divided into 3 techniques namely, Knowledge-based, analytical-based and simulation-based. All of the techniques have advantages and disadvantages and quite often, the decision is made based on the trade-off between speed and accuracy. One of the important blocks in optimisation-based approach is the optimisation technique. Some of the optimisation techniques including multi-objective optimisation were discussed in the second part of the chapter. Another important subject in analogue circuit design is the impact of process variations to the circuit performances. The last part of the chapter reviews some of the techniques that have been used to consider the process variation in the design stage and optimise the circuit yield. All the discussions in this chapter provide the fundamental understanding in the motivation behind the technique used for the work presented in this thesis.

Chapter 3

Review of Circuit Simulation and Modelling

3.1 Introduction

One of the important components for the simulation-based optimisation design technique reviewed in previous chapter is the circuit simulator. This chapter discusses the fundamentals behind circuit simulation including type of analyses involved and device modelling related to the simulator.

Computer-aided simulation is a powerful aid during the design or analysis of VLSI circuits and is considered as an essential step in the design of modern integrated circuits. In circuit simulation, a simulator is used to solve non-linear ordinary differential equations that describe the behaviour of the system. The mathematical equations that describe the component behaviour is called a model. The simulator interprets the list of individual models and construct a matrix of equations for the complete system to be solved. The most widely known and used circuit simulation program is SPICE (simulation program with integrated circuit emphasis) [73].

3.2 Analogue Circuit Simulation

Circuit simulation is a method whereby electric circuits are modelled using mathematical equations representing individual elements that to be solved to determine the function of the circuit. This section reviews the key concept involved for analogue circuit simulation.

3.2.1 Circuit Neltlist

In circuit simulation, a system is described as a list of individual models, called a netlist. The netlist provides a description of the topography of a circuit and is simply a list of elements that make up the circuit. The individual model represents all the elements in the circuit diagram. Circuit nodes are formed whenever two or more elements meet. Figure 3.1 and 3.2 show a circuit diagram for a differential pair topology and the netlist of the circuit respectively.

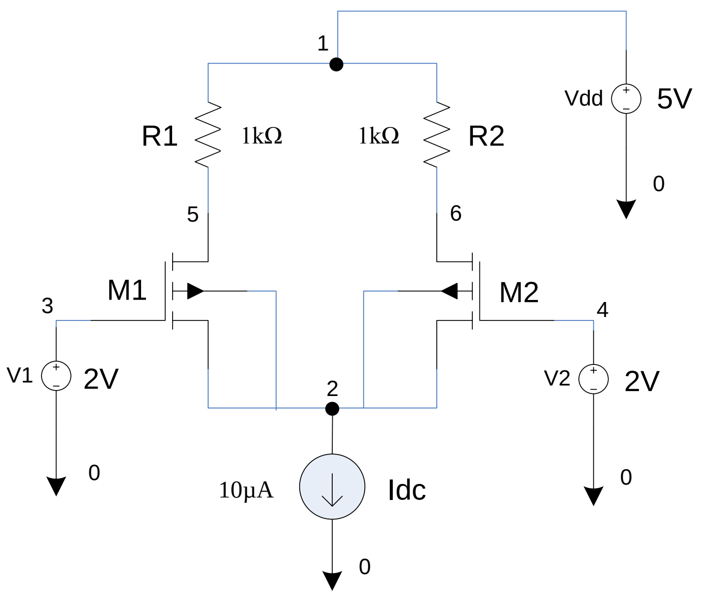


Figure 3.1 Circuit diagram for differential pair

```
VDD 1 0 5V
R1 1 5 1k
R2 1 6 1k
I1 2 0 10µ
V1 3 0 2V
V2 4 0 2V
M1 5 3 2 2 N_MOS l=2u w=10u
M2 6 4 2 2 N_MOS l=2u w=10u
.MODEL N_MOS NMOS  ( LEVEL = 1
+ KP = 20u
+VTO = 0.8V
+LAMBDA = 0.095 )
.MODEL P_MOS PMOS ( LEVEL =1
+ KP = 20u
+ VTO = -0.8V
+ LAMBDA = 0.095 )
```

Figure 3.2 Netlist for differential pair

In SPICE, the circuits are represented by a system of ordinary differential equations. These equations are solved using several different numerical techniques. The equations are constructed using Kirchhoff's voltage and current laws (KVL and KCL). KCL is used to solve the current flowing into each node. One equation is written for each node in the circuit except for ground node. Normally, the ground node in circuit netlist is numbered as zero. KVL is used to represent the voltage source or inductors elements as a function of the branch voltage in a circuit design. A loop equation based on KVL is written around each voltage source or inductor. Therefore, the total number of equations to be solved in circuit simulation is the number of nodes plus the number of voltage sources.

3.2.2 Types of Analysis

In circuit analysis, there are three types of analysis that are commonly used: DC, AC and transient analysis. DC analysis is used to examine the steady-state operation of a circuit. It tells about the voltages and currents if the inputs were held constant for an infinite time. AC analysis is used to examine circuit performance in the frequency domain and transient analysis is performed in the time domain and it is computationally intensive compared to the other two analyses.

3.2.2.1 DC Analysis

DC analysis calculates the steady-state response of a circuit (with all inductors shorted and capacitors opened). There are several analyses that can be done in this type including operating point analysis (.OP), DC solutions over the range of input condition (.DC) and small signal DC transfer function (.TF). Operating point analysis is used to determine the DC bias point (Q-point) of the circuit. .DC statement is used to sweep the specified voltage source over specified range while determining the DC bias point.

To calculate the DC solution, Kirchoff's equations need to be solved. However, due to the non-linear characteristics of the circuit elements, a non-linear solution technique such as Newton's method [47] is used. The basic Newton's method formula is given in equation 3.1 where F(X) = 0 is the equation to be solved, where both F and X are vectors of dimension N. (F is the system equations from modified nodal analysis, and X is the vector of voltages and current that are solving for). X^i is the initial value of X and X^{i+1} is the value of X a the next iteration. The term J is a NxN square matrix of partial derivatives of F, called the Jacobian [74].

$$X^{i+1} = X^i - J^{-1}.F(X^i) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.1)$$

The equation is used iteratively until the vector x converges to the correct solution. Most of the works in calculating the solution is involved in calculating J and its inverse J^{-1}. Simulator programs such as SPICE may require 50 or more iterations to achieve convergence. This is normally depends to the initial value. For a poor initial value, the convergence is not obtained until the last few iterations.

3.2.2.2 AC Analysis

AC analysis is used to calculate the frequency response of linearized behaviour of a system. The analysis is useful for calculating frequency domain function such as gain, 3db frequency, phase response and others. In this analysis, all signals are represented as a DC component, V_{dc} plus a small sinusoidal component, V_{ac}. The steps in AC analysis start with calculation of DC operating point of the circuit. A linerized circuit is constructed at this Q-point. This is done by replacing all the nonlinear elements with their linearized equations and all inductors and capacitors are replaced by complex impedances. Nodal analysis is then used to reduce the circuit to a matrix form and can be solved using Gaussian Elimination to calculate the node voltages.

3.2.2.3 Transient Analysis

Transient analysis is one of the powerful circuit analyses and justifies the benefit of circuit simulator due to the difficulity to analytically calculate the transient response of a circuit [75]. This analysis can be used to analyse many circuit characteristics in the time-domain such as distortion, switching speed, slew rate and others. It is also the most CPU intensive and takes longer simulation time compared to AC or DC analysis.

In a transient analysis, time is discretized into intervals called time steps. Typically, the time steps are of unequal length, with the smallest steps being taken during intervals where the circuit voltages and currents are changing more rapidly. The first step performed by SPICE in transient analysis is to compute the initial DC or bias point condition with the assumption of voltage across capacitors is zero, current through inductors is zero and the value for dependent sources is zero. Once the initial bias point has been calculated, iterative numerical techniques are used to obtain a solution. One example of a numerical method employ by SPICE is the Trapezoidal Method. Trapezoidal method uses one past time information to calculate the next time point solution. For example, using trapezoidal method, the current, *I* in capacitor in the next time step is given by 3.2.

$$I(t_{k+1}) = \frac{dQ}{dt} = 2\frac{Q(V(t_{k+1})) - Q(V(t_k))}{h} - I(t_k) \ldots\ldots\ldots\ldots\ldots\ldots(3.2)$$

Where *h* is the time step given by $h = t_{k+1} - t_k$. All modern circuit simulators feature automatic time step control so that the time step is allowed to be variable during simulation. This feature selects small time steps during intervals where changes are occurring rapidly and large time steps in intervals where there is little change. This will improve the efficiency of the simulation with regards to the computing power requirement.

3.3 Modelling Theory

3.3.1 Definition of a Model

In circuit simulation, a model represents physical elements of a system that are to be studied or simulated. For example, an amplifier may contain several elements and during circuit simulation, these elements are represented by their own model such as transistor model, resistor model and capacitor model. The model consists of a set of equations and parameters that characterize the exact behaviour of the physical element between the connection points. Figure 3.3 shows how a resistor can be modelled in a circuit simulation. This model represents the resistor behaviour in term of voltage and current between the connection points.

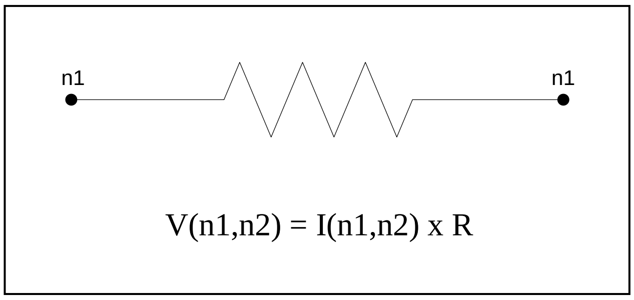


Figure 3.3: Resistor Model

The SPICE circuit simulator has a number of built-in elements such as resistors, capacitors, inductors, voltage and current sources, MOSFETs, BJTs and others. For an active element like a MOSFETs, the model contains a number of parameters that represents the transistors. This model with the set of parameters is used in a circuit simulator to simulate how a particular circuit will behave. The accuracy of the model depends on how closely the model matches the actual behaviour of the transistor.

3.3.2 Device Modelling

Active elements in a circuit, such as a transistor, contain a set of parameters that characterise the behavioural of the element. This set of parameters is called device model. A number of MOSFET device models have been provided over time with the simulator program, SPICE. This section concentrates on the standard MOS models provided by UC Berkeley's SPICE program because these models have become the standard models used by most circuit simulator programs.

3.3.2.1 MOS Levels 1, 2 and 3

These are the earliest MOS device models that come with SPICE program. Level 1 is a first order model and is rarely used. Level 2 and 3 are the extensions of level 1 model and have been used extensively [75]. Level 2 and 3 contain small number of parameters and suitable for circuit simulation down to 1μm channel length. There are a lot of limitations in these models for analogue application due to the lack of certain parameters such as G_{ds} (derivative of drain current with respect the drain voltage) and mobility degradation. Newer models have to be developed to increase the number of parameters that can accurately describe the component behaviour.

3.3.2.2 Berkeley Short-Channel Igfet Model (BSIM)

To overcome the shortcomings of level 2 and 3, the BSIM models were developed. The main difference between BSIM models and level 2 and 3 is the approach in incorporating the geometry dependence [75]. In level 2 and 3 models, the geometry dependence is built in directly into the model equations while in BSIM models, each parameter is written in terms of combination three terms given by equation 3.3

$$Parameter = Par_0 + \frac{Par_L}{L_{eff}} + \frac{Par_W}{W_{eff}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.3$$

Where Par_0 is the zero order term, Par_L is for the length dependence of the parameter, Par_W is for width dependence and L_{eff} and W_{eff} are the effective channel length and width respectively. On top of that, the number of parameters for BSIM models is larger than level 2 and 3.

The original goal for BSIM model is to fit better than level 2 and 3 for submicron channel length technology. However, the shortcomings of the early BSIM model are the inability to fit over a large number of geometry variations and there is still no Gds parameter in the model that is needed for analogue application. BSIM2 model was an extension of BSIM model that was developed to address the limitations. BSIM2 model includes parameters to model the G_{ds} in transistor and with several other modifications, BSIM2 model fit better compared to BSIM model. However, BSIM2 model comes with more than twice as many parameters as BSIM. Even with all the extension, it still does not address the problem of fitting large geometry variations faced by previous model. Due to the shortcomings of BSIM2 model, Berkeley introduced the BSIM3 model. However, BSIM3 is not an extension of the BSIM2 model, but it is entirely new model and in some sense is more related to level 2 and 3 models. BSIM3 revert back the geometry dependence into incorporating directly into the model equations as level 2 and 3 models. It is still an evolving model where it can be modified to fit better and improve the accuracy. One of the models in BSIM3 variants is BSIM3v3 and this is the type of model used in the design examples shown in this thesis.

3.3.3 Hardware Description Language (HDL) modelling

One of the advantages of HDL modelling is the capability to represents the system at various levels and is often considered as a multi-domain language. As discussed earlier, SPICE models ares used to represent a system in a circuit level which is the lowest level in the circuit design. HDL language such as Verilog-A were designed to be compatible as an extension of SPICE to represent the system at multiple abstraction level including circuit level [76].

Mathematical equations can be entered directly into Verilog-A language as well as SPICE-like circuit elements. Equations can be used to construct new models for electrical devices. Behavioural models and structural models can be constructed to model complex circuits such as op-amps, Voltage Control Oscillators, Phase Lock

Loops, etc. The behavioural simulation can be done in a small fraction time compared to circuit level simulation. With special interface elements, it is possible to connect an analogue block to a digital simulator, making mixed-mode simulation possible. The analogue behavioural capability allows the designer to span the abstraction levels, allowing direct access to the underlying technology while maintaining the capability of system-level modelling and simulation. As such, the analogue and mixed-signal system can be described and simulated at a high-level of abstraction early in the design cycle to facilitate full chip- architectural trade-offs.

In general, a system consists of interconnected components or blocks that output a response based on given stimulus or input. Verilog-A allows the system of analogue and mixed-signal to be described in terms of circuit components and modules. A structural description in Verilog-A is a description where another modules are instantiates or called within its definition.

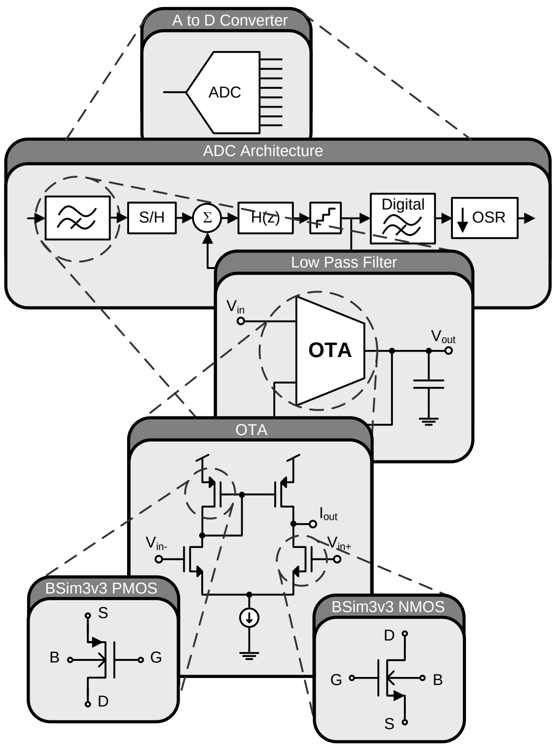


Figure 3.4: Typical hierarchy level in analogue circuit design

Structural description allows the designer to pass the parametric specifications and connections throughout the levels of hierarchy in the design. Figure 3.4 shows a typical hierarchy level in analogue circuit design. This figure shows the hierarchy design for analogue to digital converter (ADC) that consists of several levels of hierarchy moving from functional blocks to individual transistors. In hardware description language, the structural description of ADC can be made by instantiating all other modules underneath it.

In a module, analog and mixed-signal circuits can be described in a behavioural description. The descriptions in a module are the mathematical equations that mapped the input signal to the output. For example, equation 3.4 shows a behavioural description of output voltage that is described as the multiplication of the input voltage and gain parameter. Once the behavioiural model has been completely described, SPICE simulator such as Cadence Spectre and HSpice can be used to simulate the behavioural system n a similar way as circuit simulation.

$$V(out) < +V(inp) * (-gain) \ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.4)$$

3.4 Summary

The circuit simulator plays an important role in simulation-based design and it is one of the major factors that contribute to the high computational cost of the technique. The accuracy of the simulator highly depends on the model that being used during the simulation. An accurate device model will provide accurate simulation but with the expense of design speed. The computational cost is worsening for a large analogue system. Therefore, hierarchical-based design and behavioural modelling have been used to overcome this limitation. Both of the hierarchical-based design and behavioural modelling will be used for various design examples in the thesis.

Chapter 4

Yield Optimised Design

4.1 Introduction

One of the big challenges faced by analogue circuit designers in a deep sub-micron design is the process variations which cause the designed circuit to deviate from its nominal performance and thereby result in a low yield. The impact of the process variations to the analogue circuit has been discussed in chapter 2. Due to the close relationship between higher yield and higher profit, this problem has became a major concern in circuit design and led to early consideration in the design process , a technique termed as Design For Yield (DFY) [77].

The research focus for analogue integrated circuit automation often requires a trade-offs to be made between speed and accuracy. The simulation-based optimisation approach offers a great accuracy at the expense of design time while an analytical approach is fast but suffers from accuracy limitations. The same can be said for yield optimised design where an approximation based approach is fast but lacking accuracy, compared to a Monte Carlo simulation based approach which produces high accuracy results at the cost of computational time.

The complexity and variability associated with modern deep sub-micron transistor technology, has motivated this research to choose a high accuracy approach. A higher accuracy method produces a product that meets the specifications and at the same time promises a higher yield. This has also motivated the simulation-based optimisation approach that to overcome the failure of other approaches to translate the designed circuit into practical use [31]. Therefore the works presented in the remaining of this thesis are primarily based on simulation-based optimisation and Monte Carlo simulation methods.

This chapter will address the integration of yield performance parameters to the simulation-based optimisation methodology for analogue circuit design. The chapter starts with a modification made to the simulation-based optimisation algorithm to include Monte Carlo simulation as part of the design flow. This approach is compared to other yield optimisation approach in order to demonstrate the advantage given by the proposed method. In order to reduce the simulation time, the method is improved by introducing a multi-objective optimisation approach in the design flow. The improved yield optimisation methodology is then compared with NeoCircuit [10], a commercial circuit optimiser and will demonstrate the advantage of the MOO approach. The concept of yield optimised-design through Multi-Objective Optimisation and Monte Carlo simulation introduced in this chapter provides the key components to the works presented in this thesis.

4.2 Integrated yield optimised model

In yield optimised design strategy, yield is integrated as one of the performance functions. This strategy is modelled as illustrated in figure 4-1.

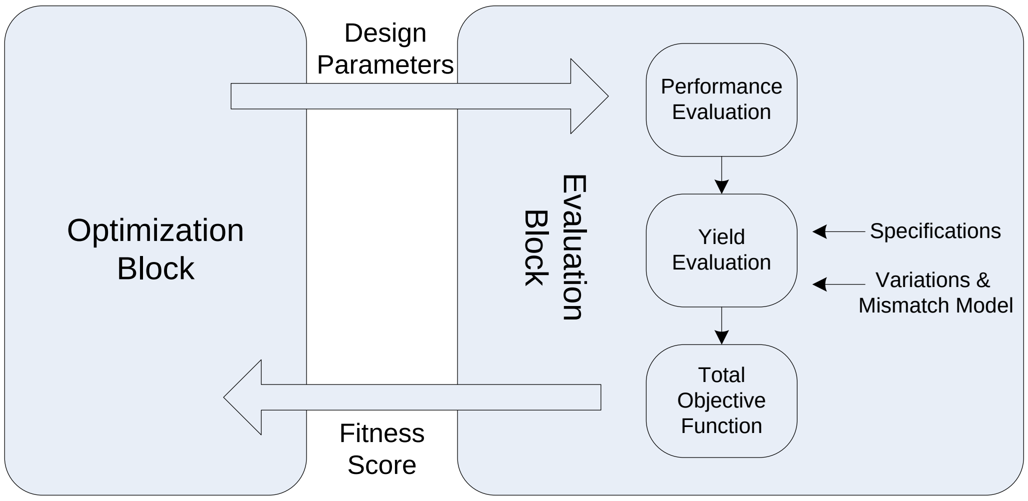


Figure 4-1: Integrated yield optimised model

The model shown in figure 4-1 is based on the simulation-based optimization approach with a small modifications in the performance evaluation block. As discussed in chapter 2, the performance evaluation block is a SPICE simulation that will simulate all the performance functions including the yield of a design. The yield is estimated using Monte Carlo simulation incorporating all the process variations and mismatch model of a particular technology. All the performance functions and yield results from the simulations are added together using a weight-summation method in order to find the total cost function. This is similar to the conversion of constrained optimisation formulation to unconstrained fashion employs by various simulation-based techiwques. The total cost function will be used by the optimizer block as the score indicator for the individuals (set of design parameters). The optimizer block will iteratively generate design parameters using Genetic Algorithm to optimize/improve the total cost function until convergence criteria is met. The convergence criteria is met when in a single generation, the mean of the total cost function closely match (within 0.5% different) with the value of the best total cost function as explained in chapter 2. At the end of the optimization, a circuit solution is found that gives the best trade-offs among the performance function and at the same time able to achieve higher yield.

4.3 Design Example for Yield Optimised Model

This section demonstrates the model introduced previously with a simulation of circuit example. The proposed method was applied to a Symmetrical-OTA circuit topology. The OTA was chosen as the case study because it is a fundamental block that is widely used in numerous analogue circuit design applications.

4.3.1 OTA design and objective functions

The chosen circuit topology is shown in figure 4-2. It consists of differential input, current mirror and single ended output stage. Transistor pair M1, M2 is a current mirror that provide the current source for differential input pair M4,M5. Drain current of M4 is mirrored to drain of M9 by current mirror pair M7,M9 and drain current of M5 is mirrored to drain of M6 by current mirror pair M10,M8 and M3,M6. Since a matching transistor size is very important in differential pair and current mirror, all the transistors are grouped as pairs. This is to ensure the size of the transistor generated by the optimizer is same for both of the transistor in the pair.

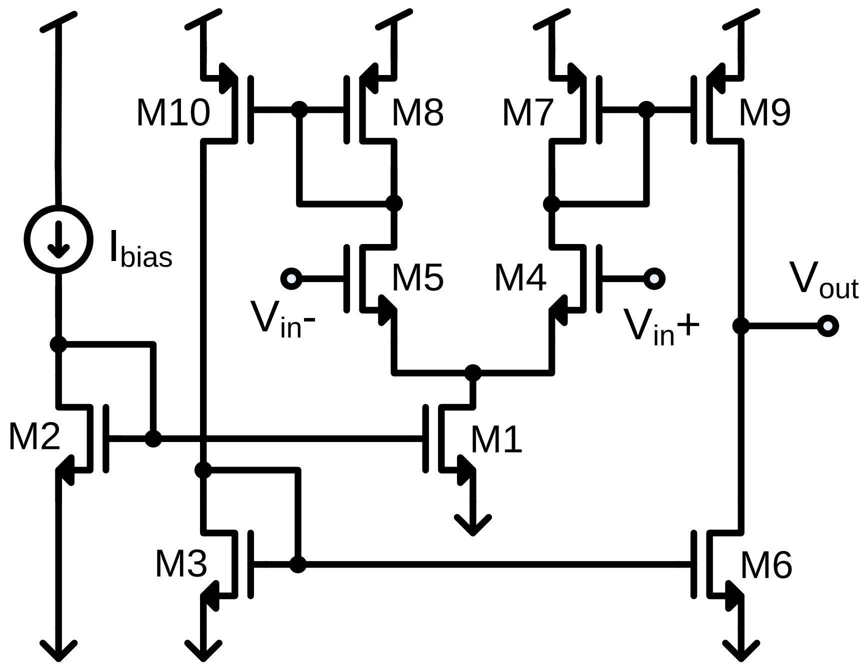


Figure 4-2: Symmetrical OTA topology

In this example, there are 4 transistor pairs that need to be sized make up a total of 8 designable parameters. Transistors M1 and M2 in this example are fixed since this is simply a mirror for the current source. There are 8 performance functions to be optimized including the overall yield. Table 4-1 shows the performance functions and their specifications.

Performance function:	**Specification:**
Open Loop Gain	> 50db
Phase Margin	> 60 deg
GBW	>15 MHz
Voltage Offset	< 15mV
Slew Rate	> 15 V/µs
Power	Minimized
Area	Minimized
Yield	Maximized

Table 4-1: Performance functions and specifications

The designable parameters are constrained to a reasonable range so that the total area of the design will not exceed 2mm^2 in size. This defines the decision space of the optimisation. The range of the designable parameters is shown in table 4-2.

Design Parameter:	**Range:**
W_1 (M5,M4)	10um - 60um
L_1 (M5,M4)	0.12μm - 4μm
W_2 (M7,M9)	10um - 60um
L_2 (M7,M9)	0.12μm - 4μm
W_3 (M10,M8)	10um - 60um
L_3 (M10,M8)	0.12μm - 4μm
W_4 (M3,M6)	10um - 60um
L_4 (M3,M6)	0.12μm - 4μm
W_{g1}-W_{g8} (weight)	0.1 – 1.0

Table 4-2 Design Parameters

As mentioned earlier in this chapter, the total cost function is calculated using weight-summation method. The weight valules for the summation are determined by the optimizer block. Therefore in this algorithm, the optimizer (Genetic Algorithm) will not only generate the designable parameters but also the weight for the performance function. In table 4.2, W_{g1}-W_{g8} are all the weights for the performance functions. Each individual generated by the GA will consist of a set of designable parameters for the circuit and weight values for the performance function as defined by the GA string. Figure 4-3 shows the construction of the GA string for this example.

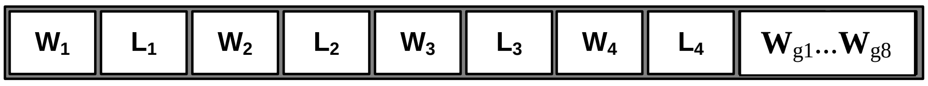


Figure 4-3: GA String

Once the GA string for the optimization has been constructed, the optimization will start with a random set of designable parameters. The design parameters generated by GA will be used to replace the parameters in SPICE netlist for the performance simulation. For the yield estimation, a Monte Carlo simulation with 200 samples is used for all of the performance functions. Based on the specification, the yield of the individual performances is calculated. The yield for the individual performance is compared to determine the overall yield of the design. All the results from the evaluation of 8 performance functions are multiplied with their respective weights and are summed together to determine the overall cost function. The objective of the optimisaton is to maximise the total cost function. For minimisation type performance, for example voltage offset, the performance is multiplied with -1 in order to convert it into maximisation formulation. From one generation to another, GA will try to maximise the cost function which in turn will maximise/minimise all the performance functions accordingly. The optimisaton process is repeated until the convergence criteria is met. Once the criteria is met, the optimization is stopped and the result is a design that gives the best performance trade-offs and higher yield.

The convergence criteria is met when the mean(average) of the cost function in a generation closely match the maximum cost function of the generation. Maximum cost function is the best individual with the highest fitness score in the generation. The average fitness score in the generation is calculated and compared with the best individual. Once the mean fitness score closely match to the max fitness score, the optimization is said to converge. Figure 4-4 shows the convergence of the optimization that is achieved after 30 generations.

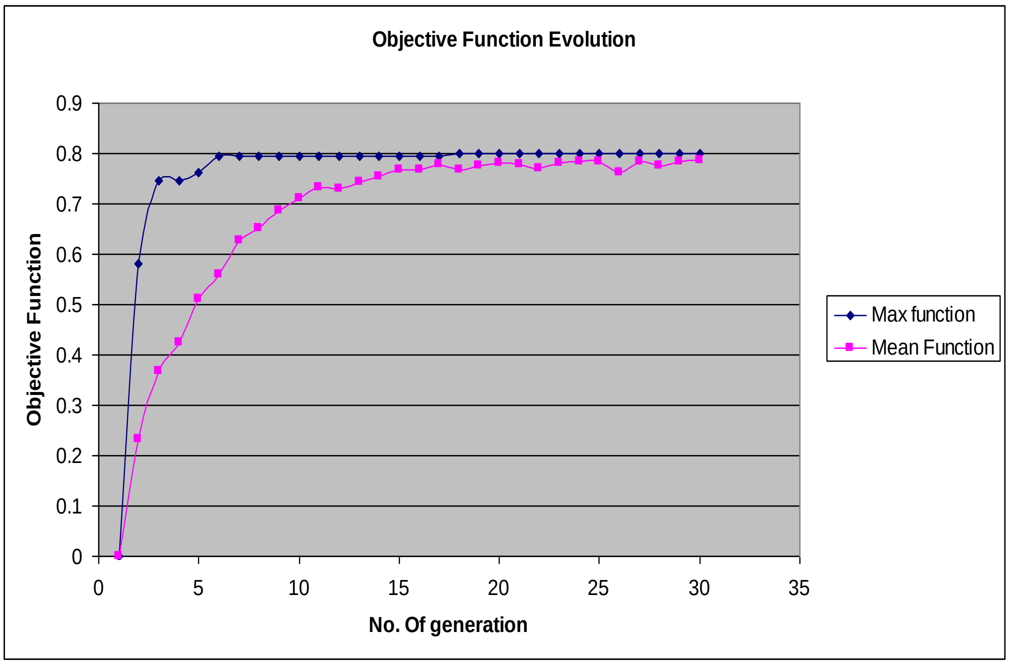


Figure 4-4: Convergence Criteria

4.3.2 Comparison With Design Centering Approach

One of the benefits of integrating yield as one of the performance functions is the ability to optimize the yield with respect to the trade-offs of the performance functions. In this way, the optimization of the performance functions is balanced between each other in order to avoid excessive performance in some of the objective functions that can limit the overall yield. To show the advantage of the approach, a comparison is made with design centring method. As described in chapter 2, design centering is an indirect method for yield optimization that attempt to place the nominal design at the centre of the acceptability region. In such attempts, all the performance functions will be pushed as far as possible from the boundary (specification) to maximized the yield. Table 4-3 shows the comparison result.

		Yield-Optimised Approach		**Design Centring Approach**	
Performance Function	**Spec**	**Result**	**Indiv. Yield**	**Result**	**Indiv. Yield**
Gain	> 50dB	50.7 dB	100%	50.9 dB	100%
Volt. Offset	< 15mV	7.5 mV	89%	10.77 mV	72%
GBW	> 15 MHz	16.67 MHz	96%	17.08 MHz	100%
Phase Margin	> 60 deg	68 deg	94%	69.8 deg	100%
Slew Rate	> 15 V/us	16.1 V/us		17.7 V/us	
Power	Minimised	256.2 uW		255.7 uW	
Area	Minimised	209.3um²		195.3um²	
CPU Time		**2h 40m**		**1h 05m**	

Table 4-3: Simulation result and comparison

As can be seen from table 4-3, with yield as a performance function, the optimization is targeted towards the trade-offs among the competing objectives similar to multi-objective optimisation approach. In the design centring approach, there are 3 performances (gain, GBW and phase margin) that achieve 100% yield. However with such performances, the improvement/optimisation for voltage offset is limited and becomes very low and might affect the overall yield. This observation leads to the consideration of multi-objective optimisation technique in the yield-optimised approach and become the key component in the methodology presented in this thesis.

4-4 Improved yield optimised algorithm

The method proposed in the previous section shows the improvement that can be achieved compared to traditional yield optimization approach. However, the limitation of this approach is high CPU runtime. This is due to the Monte Carlo simulation that need to be run for each design sample during the optimization. In this section, this issue is taken into consideration to reduce the design time. There are two important modifications in the approach: first, instead of searching for single optimum solution, a set of optimum solutions that is called pareto-points are explored. This is done by running multi-objective optimization using WBGA to obtain the Pareto-front. The concept of Pareto has been explained in chapter 2. Second, Monte Carlo simulation will only need to be applied on a set of solutions in the feasible region that is defined by the performance specification. This reduces the number of Monte Carlo simulation significantly and thus reduces the overall design time. Figure 4-5 shows the design flow for the improved algorithm.

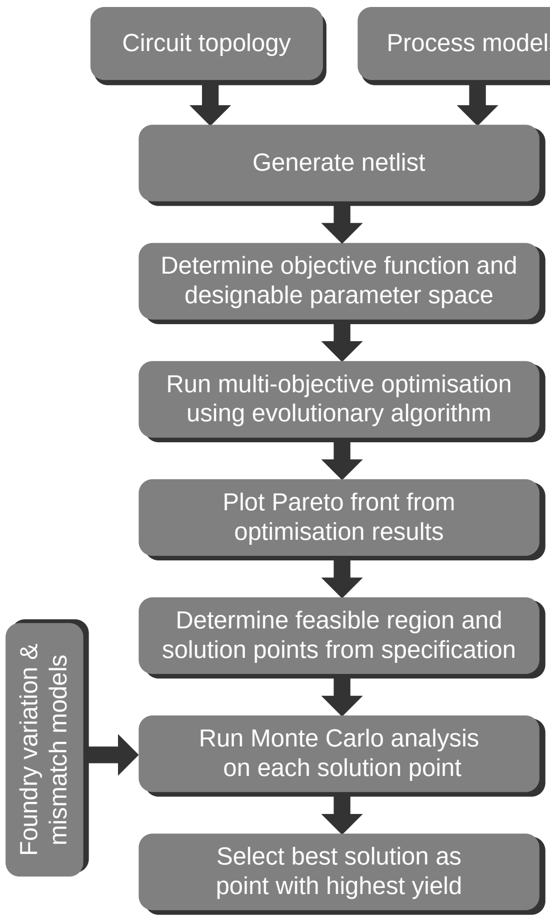

Circuit topology
Process models
Generate netlist
Determine objective function and designable parameter space
Run multi-objective optimisation using evolutionary algorithm
Plot Pareto front from optimisation results
Determine feasible region and solution points from specification
Foundry variation & mismatch models
Run Monte Carlo analysis on each solution point
Select best solution as point with highest yield

Figure 4-5: Yield targeted algorithm

In multi-objective optimization, where multiple conflicting objectives are important, there generally will not be a single optimum solution that optimizes all the objectives. The optimization will result to a number of optimal and non-optimal solutions. It is necessary at this point to determine the Pareto front which consists of the most optimal, non-dominated solutions in the objective space. The solution points on the Pareto-front is the optimal solution that gives the best trade-offs among the competing objectives.

Once the Pareto-front has been obtained, the specifications can be added to the plot. This will result to a small region defined by the specifications that is called feasible region. This region contains all the solutions that meet the specifications. However, due to the statistical variations, the solutions on this region may still fall below specification when fabricated. In order to find the solution that will give high overall yield, Monte Carlo simulation is done on all the solution points on the Pareto-front in this region. Compared with previous example, this approach requires far fewer Monte

Carlo simulations due to the small number of solutions in the feasible region, mitigating the computational overhead. Once the Monte Carlo simulation for all solution points completed, the solution that gives the highest yield is then selected as the best solution for the design.

4.5 Design Example for Improved Yield Optimised Algorithm

This section demonstrates the newly proposed algorithm with the same example shown in figure 4-2. For illustrative purpose, performance objective is reduced to two functions, Open loop gain and Phase Margin. The number of designable parameters and GA string construction is same as previous example. The specifications for this example are shown in table 4-4.

Objective function:	**Specification:**
Open loop gain	>50dB
Phase margin	>74deg
Area	minimized
Power	minimized

Table 4-4: Design specifications

4.5.1 Pareto front and feasible region

Multi-Objective optimisation (WBGA) was applied to the design example and from this, the objective space of the optimisation has been plotted. Figure 4-6 shows the plot of the objective space for open loop gain and phase margin and its Pareto-front. All the solutions lie on this front are the optimal solutions that best describe the trade-offs of the objectives. To find the feasible region of the design, specifications line for both of the performance functions are inserted in the plot.

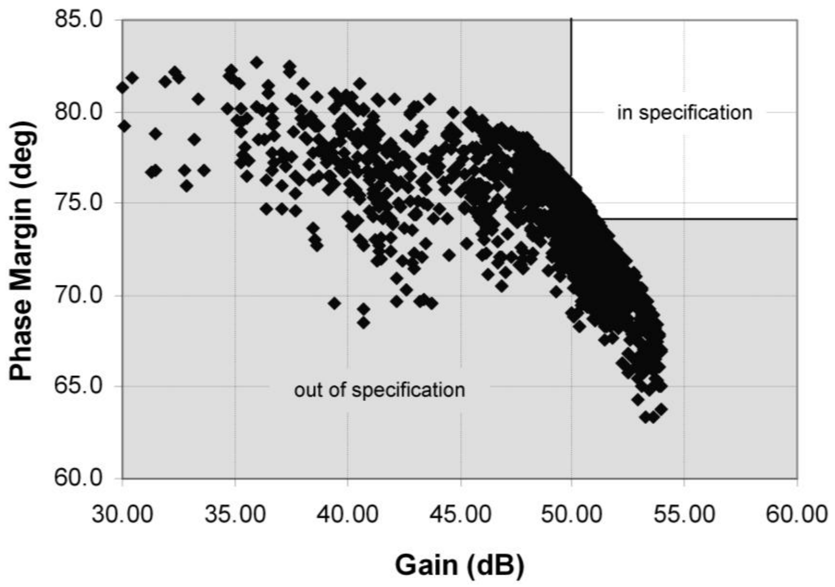


Figure 4-6: Objective space and Pareto-front

The in-specification area shown in figure 4-6 narrows down the solution space into small feasible region. This region is shown in detail in figure 4-7. It can be seen from this figure, that there are only 10 optimal solution points on the Pareto-front of the region as labelled by the number. These are the points that will be used in the next step of the algorithm to determine the best solution that gives high yield.

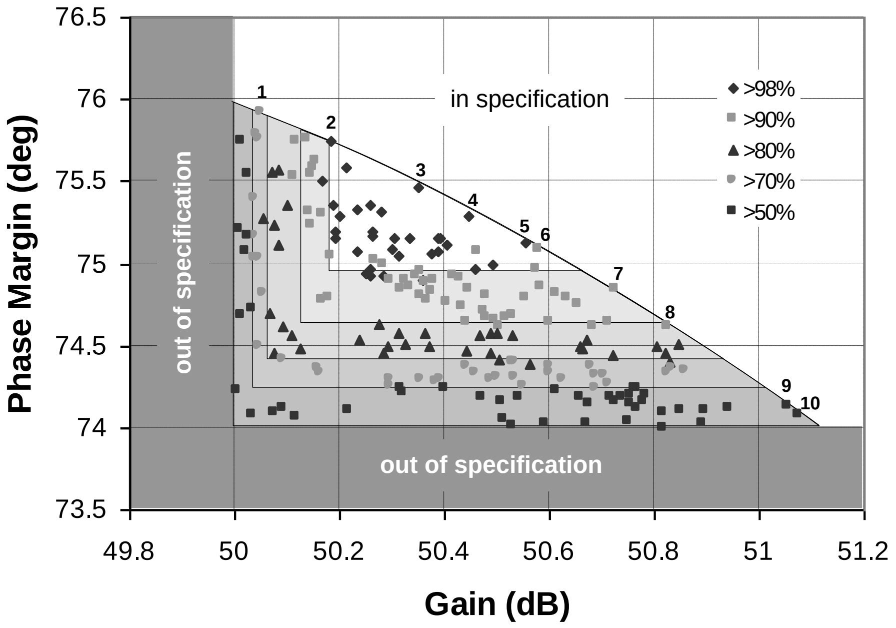


Figure 4-7: Detail view of feasible region

4.5.2 Monte Carlo simulation

All the optimal solutions within the feasible region undergo a Monte Carlo simulation using foundry process variations and mismatch model. Some examples of the parameters used during the simulation are shown in Figure 4.8.

```
*---------------------------------------------
*                nmoshs
*---------------------------------------------
.param nmoshs_vth0 = '0.13+0.5*0.015*nsigma_nmoshs_vth0'
.param nmoshs_dmu = '0.0+0.5*5e-2*nsigma_nmoshs_dmu'
.param nmoshs_drdsw = '0.0+0.5*5e-2*nsigma_nmoshs_drdsw'
.param nmoshs_dcjb = '0.0+0.5*10e-2*nsigma_nmoshs_dcjb'
.param nmoshs_dcjgate = '0.0+0.5*20e-2*nsigma_nmoshs_dcjgate'
.param nmoshs_dcjsw = '0.0+0.5*20e-2*nsigma_nmoshs_dcjsw'
.param nmoshs_djsdbr = '0.0+0.5*1*nsigma_nmoshs_djsdbr'
.param nmoshs_djsdgr = '0.0+0.5*1*nsigma_nmoshs_djsdgr'
.param nmoshs_djsdsr = '0.0+0.5*1*nsigma_nmoshs_djsdsr'
.param nmoshs_djsgbr = '0.0+0.5*1*nsigma_nmoshs_djsgbr'
.param nmoshs_djsggr = '0.0+0.5*1*nsigma_nmoshs_djsggr'
.param nmoshs_djsgsr = '0.0+0.5*1*nsigma_nmoshs_djsgsr'
.param nmoshs_rstir = '4000+0.5*1200*nsigma_nmoshs_rstir'
.param nmoshs_rstil = '0+0.5*0*nsigma_nmoshs_rstil'

*---------------------------------------------
*                pmoshs
*---------------------------------------------
.param pmoshs_vth0 = '-0.19056+0.5*0.015*nsigma_pmoshs_vth0'
.param pmoshs_dmu = '0.0+0.5*5e-2*nsigma_pmoshs_dmu'
.param pmoshs_drdsw = '0.0+0.5*5e-2*nsigma_pmoshs_drdsw'
.param pmoshs_dcjb = '0.0+0.5*10e-2*nsigma_pmoshs_dcjb'
.param pmoshs_dcjgate = '0.0+0.5*20e-2*nsigma_pmoshs_dcjgate'
.param pmoshs_dcjsw = '0.0+0.5*20e-2*nsigma_pmoshs_dcjsw'
.param pmoshs_djsdbr = '0.0+0.5*1*nsigma_pmoshs_djsdbr'
.param pmoshs_djsdgr = '0.0+0.5*1*nsigma_pmoshs_djsdgr'
.param pmoshs_djsdsr = '0.0+0.5*1*nsigma_pmoshs_djsdsr'
.param pmoshs_djsgbr = '0.0+0.5*1*nsigma_pmoshs_djsgbr'
.param pmoshs_djsggr = '0.0+0.5*1*nsigma_pmoshs_djsggr'
.param pmoshs_djsgsr = '0.0+0.5*1*nsigma_pmoshs_djsgsr'
.param pmoshs_prwb = '-0.18544+0.5*0.133*nsigma_pmoshs_prwb'
.param pmoshs_rstir = '2200+0.5*660*nsigma_pmoshs_rstir'
.param pmoshs_rstil = '0+0.5*0*nsigma_pmoshs_rstil'
```

Figure 4.8: Process variation parameters

The variation in these parameters such as threshold voltage (V_T) and sheet resistance come from the variation in the fabrication process such as oxide thickness and diffusion depths. For example, the threshold voltage can vary due to the changes in oxide thickness, polysilicon impurity levels and surface charge. All the process-specific information including the parameters, statistical variations and the transistor model are provided by the foundry in a process design kit which is part of the model file in the Cadence Spectre environment. During the Monte Carlo simulation, the process parameters are randomly changed according to the statistical variation to imitate the actual fabrication process. As explained earlier in this chapter, the Monte Carlo simulation consumes higher CPU time, but in this example, due to the small number of solution points (10 points), the simulation time is reduced significantly. The Monte Carlo simulations for all the optimal solution points were done with 500 samples and the yield percentage is calculated. Table 4-5 shows the 10 optimal solutions in the feasible region and their yield percentage.

Design Point:	**Gain (dB):**	**Phase Margin (deg):**	**Yield (%):**
1	50.17	75.8	98
2	50.35	75.5	100
3	50.45	75.3	99
4	50.54	75.2	98
5	50.57	75.1	97
6	50.72	74.9	94
7	50.81	74.6	91
8	50.86	74.5	88
9	51.04	74.2	58
10	51.06	74.1	55

Table 4-5: Design point yield percentage

From the table, the yield spread from 55% to 100% highlights the benefit of the proposed technique. For without knowledge of the yield for these optimum solutions, a designer may unwittingly choose a poor design point. From this result, design point number 2 is the best design that will produce highest yield with the process variations and mismatch during the fabrication process. By concentrating only on the feasible region for the yield estimation, the computational overhead is reduced and the entire design cycle for this example took only 48 minutes on a 1.2GHz Ultra Sparc 3 workstation.

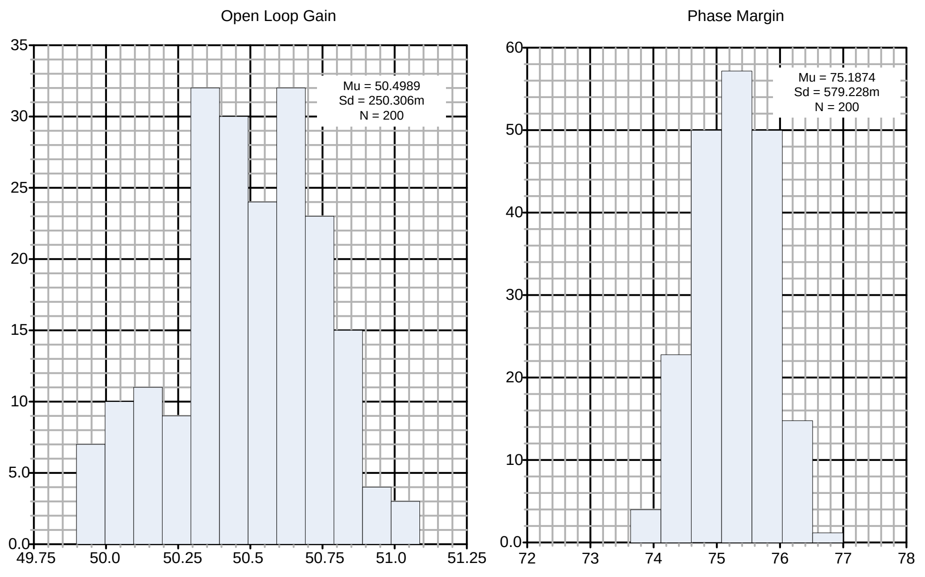
4.5.3 Comparison with NeoCircuit^{tm} Tool

To demonstrate the advantage of Pareto based optimization over conventional simulation-based approaches, a comparison has been made using the same example with NeoCircuit, a commercial optimization tool that optimizes circuit performance and yield. The tool is based on a global optimization approach that combines evolutionary and simulated annealing algorithms. The approach starts with performance optimization to meet a given specification and is followed by yield maximization to push the design far from the specification boundaries. Since there is no Pareto type exploration in the algorithm, a penalty scheme is used to reduce instances of excessive performance that may occur during yield maximization in order to maximize overall yield. This involves several iterations during the yield maximization. For example, during the first iteration, a performance function *f1* might be overdesigned and cause the optimization on performance *f2* to be limited hence resulted to a low yield. In order to increase the yield, the performance *f2* must be improved which means the performance of *f1* must be reduced. Several stages of iteration are required in order to maximize the overall design yield.

Pareto-based optimization uses a different approach where all the design performances are represented as a trade-off to make it easier to select a more balanced solution and maximize the yield. Table 4-6 summarizes the comparison between NeoCircuit tool and the proposed design methodology with the Monte Carlo histogram shown in Figure 4-9. It can be clearly seen that the Pareto-based yield optimization method performs significantly faster and produces better results than the NeoCircuit optimization. In this comparison, the Pareto-front technique completed the optimization in 48 minutes and produced a 98% overall yield whilst NeoCircuit took 1hr 29 minutes and produced a 96.5% overall yield. The comparison with NeoCircuit is useful as a “benchmark” to establish that the proposed method is at least as good as and at least as fast as NeoCircuit. However, the real benefit will become apparent later on when the Pareto-based optimisation is used to model the performance and variation of an analogue circuit and when the hierarchical-based optimisation is undertaken for system level design. Hierachical-based optimisation will be explained in the next chapter.

Parameters:	Pareto-based optimization:	NeoCircuit
Gain	50.58 dB	50.14 dB
Gain Yield	99%	96.5%
PM	75.14 deg	75.24 deg
PM Yield	98%	98%
Overall Yield	98%	96.5%
CPU Time	48 minutes	1hr 29 minutes

Table 4-6: Yield optimised design comparison



a)

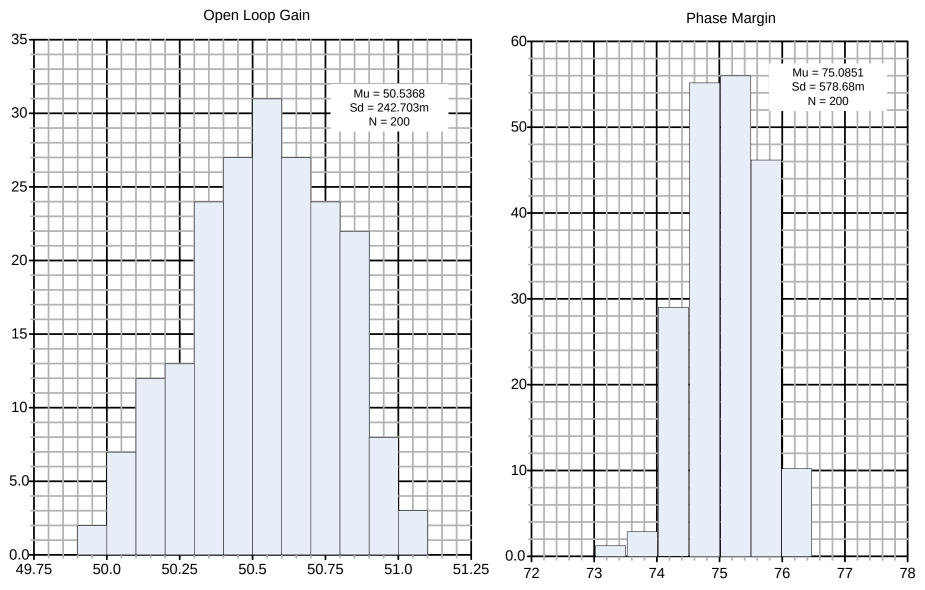


Figure 4-9: Monte Carlo histogram for gain and phase margin. a) NeoCircuit. b) Proposed methodology

4.6 Summary

In this chapter, a yield optimised design methodology has been introduced. In the proposed design model, yield integration to the optimization loop has been investigated as a method of exploring the trade-offs between the performance objectives. An example has been shown to demonstrate the benefit of yield-optimised approach compared to design centring method with 17% improvement in overall yield. However, this improvement comes with one drawback, CPU runtime. Due to the Monte Carlo simulation for all of the solutions in the objective space, the total design time becomes very high. Therefore an improvement is proposed to overcome this problem using Multi Objective Optimisation and feasible region Monte Carlo simulation.

In the new improved algorithm, a concept of Pareto-front and feasible region were introduced. Pareto-front is the outcome of a multi-objective optimization that tells the best optimal solution's trade-offs between the objective functions. Once the Pareto-front has been determined, a feasible region is defined based on the performance specifications. With such feasible region, the number of Monte Carlo simulation needed to find the yield is reduced hence, reduced overall design time. An example has been shown to demonstrate the new yield targeted algorithm that manage to reduce the design time significantly and a comparison with NeoCircuit optimiser tools shows the advantage of the proposed approach.

Chapter 5

Performance and Variation Modelling

5.1 Introduction

The first part of this thesis has introduced the concept of simulation-based design technique for analogue design automation. This approach has been used as the basis for the yield optimization algorithm proposed in chapter 4. That chapter has demonstrated the capability of multi-objective optimisation combined with Monte Carlo simulation to optimise for performance and yield. Other than high accuracy result associated with simulation-based technique, this approach creates a wholly new opportunity for circuit modelling. This is due to the high number of simulated samples that can be obtained from the optimization process. With such number of design samples, a performance model that relates the design parameters with the performance functions can be created. The idea of performance and variation modelling from multi-objective optimisation result will be presented in this chapter.

The use of simplified macromodels for analogue circuits to accelerate and enhance design exploration has a long history in mixed-signal design [78, 79]. The earliest techniques for macromodel construction relied on design expertise to create a simplified circuit model, and analytical equations needed to map the performance of the full circuit into parameters for the macromodel. More recent techniques combine the design expertise of the model structure with curve fitting method to fit macromodel parameters from samples of the full circuit's performance obtained from simulation. [80] Proposed a neural network-based methodology for creating models for estimating the performance parameters of CMOS operational amplifier topologies. This model is used together with genetic algorithm-based circuit synthesis system that demonstrates the efficiency of the performance models in operational amplifier design.

The introduction of standardized behavioural description languages offers designers the ability to mix device-level models, behavioural model and digital blocks all in the same simulation environment. Behavioural models capture the overall functionality of the circuit in terms of equations or simple circuit elements that are faster to simulate compared to the complete transistor level. Some of the concepts in circuit modelling and behavioural modelling have been described in chapter 3.

In this chapter, a behavioural modelling method is used together with the simulation-based technique to create a performance and variation model for analogue integrated circuit. The behavioural model is very helpful in a large system design where the CPU runtime often become one of the drawbacks in simulation-based approach. The idea is to use the Pareto-front from a multi objective optimization to capture the performance and variation behaviour of a circuit. Behavioural description language is then being used to implement this model that can be used for system level circuit design.

5.2 Pareto-front modelling

Pareto-front from a multi-objective optimisation represents the best trade-offs between the performance functions across the whole design space. Pareto-front modelling has been used previously for analogue circuit design [81], but most of the models do not include variation behaviour, hence are not suitable to predict the yield of the design. In this chapter, Pareto-front modelling that is capable to model the performance and the variation is proposed making it a suitable solution for a robust design technique for analogue circuit design. Figure 5.1 shows a complete design flow for the proposed methodology.

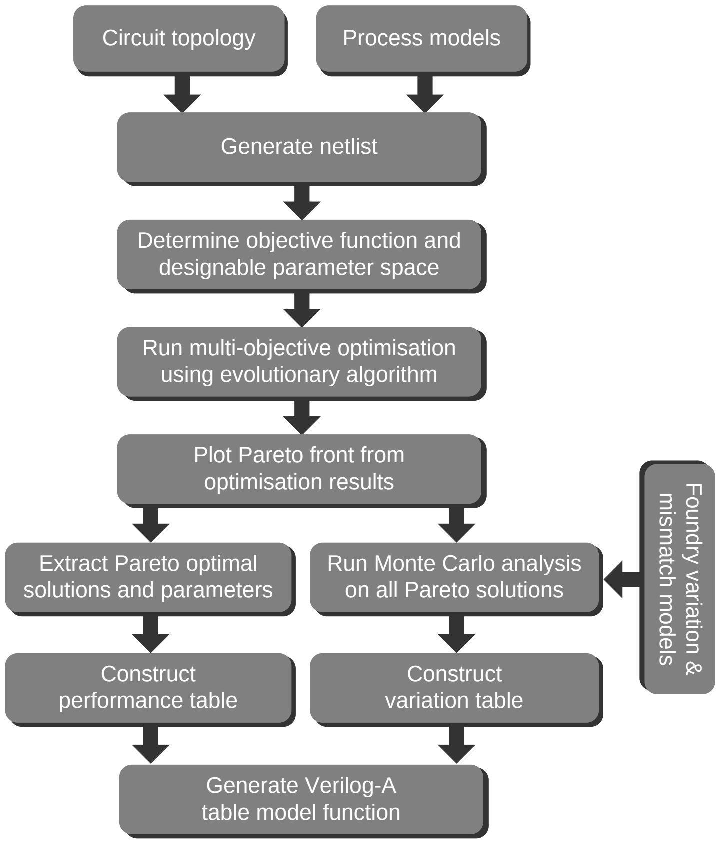

Circuit topology
Process models
Generate netlist
Determine objective function and designable parameter space
Run multi-objective optimisation using evolutionary algorithm
Plot Pareto front from optimisation results
Extract Pareto optimal solutions and parameters
Run Monte Carlo analysis on all Pareto solutions
Foundry variation & mismatch models
Construct performance table
Construct variation table
Generate Verilog-A table model function

Figure 5.1: Performance and variation's model development flow

5.2.1 Pareto-front modelling – performance

The performance model of a circuit design is a model that relates the performance of a circuit with its design parameters. In multi objective optimization, the parameter space is explored to find a solution for a circuit problem. The solution space (objective space) shows all the possible solutions that corresponding to the parameter space. The optimal performance trade-offs are represented by a Pareto-front. To model the performances, the solutions on the Pareto-front (optimum solutions) are taken and the design parameters corresponding to these solutions are recorded. All this information is stored in a text file and represents the performance model of the circuit. The model can be used to design the circuit for any design requirements that related to the modelled objective space.

5.2.2 Pareto-front modelling – variation

The Pareto-front gives optimum solutions for a circuit design. However, the solution points do not tell how the design will behave under process variations. Even though the points on the Pareto-front are the best optimal solutions, but with process variations, these performances may still fail the specifications. Therefore if a design is chosen from this Pareto-front for particular specifications, it may still result in a low yield. Variation modelling on the Pareto-front solutions can be used to observe the behaviour of the performances under process variations. As a result, a solution taken from both of the performance and variation model will meet the specifications and at the same time can provide information regarding the yield that can be expected from the design.

In order to model the variations, a Monte Carlo simulation using process variation and mismatch model is applied to all of the solution points on the Pareto-front. A standard deviation from the Monte Carlo result is calculated and a 6-sigma range ($\pm 6\sigma$) is estimated. The minimum and maximum values of the 6-sigma range are taken as the variation for the performance. The variations for all the Pareto performances are stored in a text file and represent the variation model of the circuit.

5.2.3 Interpolation from a lookup table

All of the data stored in the text file (performance and variation data) can be implemented as a lookup table using a behavioural description language. Verilog-A supports a function called table_model() function that represent a set of data points from a lookup table. This function allows the module to approximate the behaviour of the system by interpolating between the sampled data points. The syntax for this function is given in equation 5-1.

$table_model(input variables, “table file”,control string); 5-1

Where the input variables are the independent variables of the model, table file is a text file that contains the sample points of the model and the control string determines the interpolation and extrapolation method. The control string must be provided for each independent variables used in the function. There are three types of interpolation setting (1,2 or 3) and extrapolation setting (C,L or E) that can be defined using the control string indicating the chosen interpolation and extrapolation method. However, in the presented work, no extrapolation is used in order to avoid approximation of the data beyond the sampled data points that may affect the accuracy of the result. An example of the table model function including the data file is shown in section 5.3.2. With this function, the model for performance and variation can be developed behaviourally and can be used as a part of behavioural description for a larger system design. The table model approach has been used previously for modelling electrical characteristics of microelectronic devices in [82].

Interpolation is a method to connect discrete data points in a plausible ways to get a reasonable estimate data point [83]. Interpolation takes into account all the data points on the curve. The accuracy of the table model is influenced by several factors including the type of interpolation and the number of samples in the table.

Table model function of Verilog-A uses spline interpolation to interpolate new data points. Spline interpolation uses low degree polynomials that are fast and less error compared to polynomial interpolation. The principle behind spline interpolation is to divide the interpolation interval into small subintervals. Each of these subintervals is interpolated by using up to a third-degree polynomial. With a low degree polynomial, the problem of Runge's phenomenon can be avoided. Runge’s phenomenon is a problem that occurs when using high degree polynomial for interpolation where the error between the interpolating polynomial and the function grow without bound. Due to this phenomenon, at the interpolating points, the error between the points and the actual function points is small, but at the gap between the interpolating points, the error is big. Verilog-A support three type of interpolation: linear spline, quadratic spline and cubic spline interpolation.

5.2.3.1 Linear Spline

Linear spline interpolation is the simplest form of interpolation which deals with a spline that consists of first-degree polynomials. This is equivalent to linear interpolation. Linear spline interpolation is quick and easy but provides low precision results. The higher the distance between the data points, the higher the error of the interpolation. Here, the number of data points is very important to maintain the accuracy of the interpolation. Linear spline interpolation can be defined as

$$S_i(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i) \qquad 5\text{-}2$$

Generally, linear spline interpolation interpolates data from two consecutive data points. Between the data points, the slope changes abruptly and not smooth. This limitation which affects the accuracy of the interpolation can be improved by using quadratic spline or cubic spline interpolation.

5.2.3.2 Quadratic Spline

In a quadratic spline, a quadratic polynomial approximates the data between two consecutive points. for a given data points $(x_0, y_0), (x_1, y_1)...(x_{n-1}, y_{n-1}), (x_n, y_n)$, the quadratic splines are given by

$$S(x) = \begin{array}{l} a_1x^2 + b_1x + c_1 ....x_0 \le x \le x_1 \\ a_2x^2 + b_2x + c_2 ......x_1 \le x \le x_2 \\ \\ a_nx^2 + b_nx + c_n ......x_{n-1} \le x \le x_n \end{array} \qquad 5\text{-}3$$

From the above equations, there are 3n coefficients for splines: a, b and c. To solve for these coefficients, 3n equations are needed. From two consecutive data points, 2n equations can be derived. In order to get one more equation, an assumption must be made. The first spline can be assumed linear. Therefore the coefficient for a_1 can be made 0. Even with quadratic spline, the curve is not smooth enough. For this reason a third degree polynomials for each of the subinterval data points are often used to interpolate the data points.

5.2.3.3 Cubic Spline

In a cubic spline, the piece-wise interpolation curve is constructed by using third degree polynomials for each of the subinterval points. Cubic spline polynomial can be defined as

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \quad \text{for} \quad x \in [x_i, x_{i+1}] \qquad 5\text{-}4$$

Since there are n intervals for $i = 0,1,...n$ and 4 coefficients, 4n parameters are required to define the spline. One of the requirement of this spline is that the cubic polynomial to match the values of the table at both end of the intervals. This gives two conditions for each of the intervals: $S_i(x) = y_i$ and $S_i(x_{i+1}) = y_{i+1}$.These result in a continuous piece-wise function.

To make the interpolation as smooth as possible, the first and second derivatives must also be continues:-

$$\begin{aligned} S'_{i-1}(x_i) &= S'_i(x_i) \\ S''_{i-1}(x_i) &= S''_i(x_i) \end{aligned} \qquad 5\text{-}5$$

Table model function of Verilog-A allows the module to approximate the behaviour of a system by interpolating between user-supplied data points. The set of data points is stored in a text file and will be called by verilog-A module during simulation. Other than interpolation, this function can also be used to extrapolate a new data point. However, extrapolation can be inaccurate and is avoided in the presented work. The interpolation type can be selected by inserting the interpolation degree in the table model function statement as shown in Table 5.1

Interpolation Char.	Description
1	Linear Spline (degree 1)
2	Quadratic Spline (degree 2)
3	Cubic Spline (degree 3)

Table 5.1: Interpolation degree for table_model function

5.3 Modelling Example

The OTA is a fundamental building block, often employed in analogue circuit applications such as filters. This section presents a complete design example for performance and variation modelling using two different topologies for an operational transconductance amplifier (OTA) circuit: symmetrical OTA and Miller-OTA. The symmetrical OTA topology shown in figure 5-2 was used in the chapter 4 for the integrated yield optimisation example. Figure 5-3 shows the topology of the Miller-OTA. All the simulations were performed using the industry standard Cadence Spectre simulator with foundry level BSim3v3 transistor models from a standard 0.12um CMOS process technology.

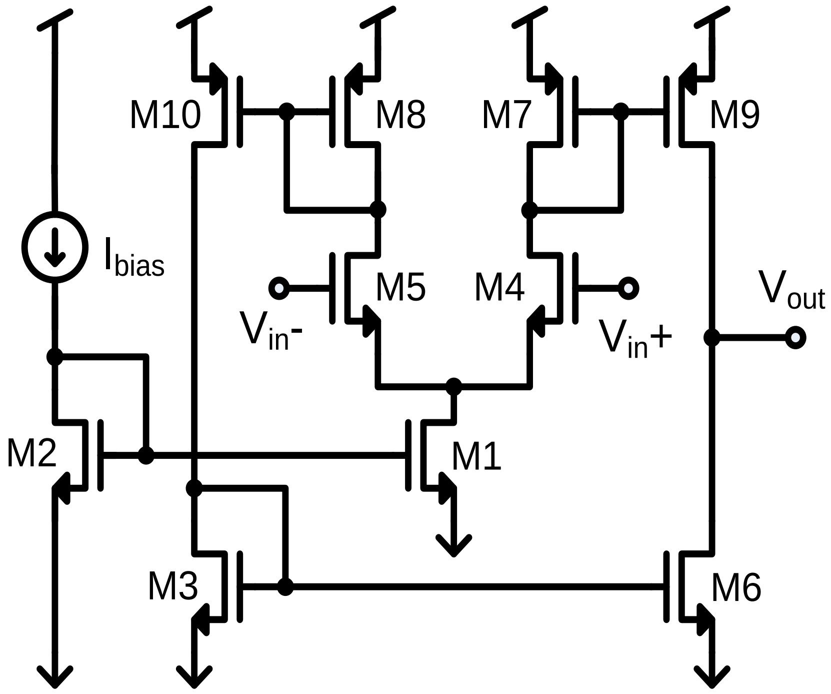


Figure 5.2: Symmetrical OTA topology

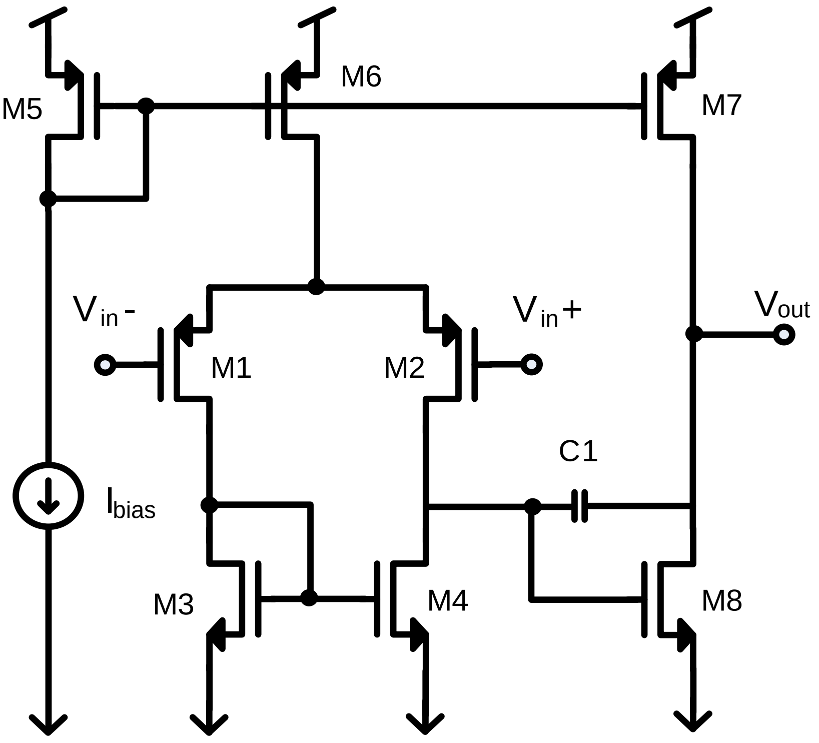


Figure 5-3: Miller-OTA topology

All transistor lengths and widths for the circuits are the designable parameters and two objective functions were chosen for this example: open loop gain and phase margin. The designable parameters are constrained within a reasonable range. All transistor lengths were specified to be between 0.12um and 4um and transistor widths were specified to be between 10um and 60um. These ranges were chosen so that the design area will not exceed the targeted transistor active area of 2mm^2. For the purpose of performance evaluation, a test-bench netlist must be created for each of the objective functions. A multi objective optimization using genetic algorithm was carried out to maximize both of the objective functions.

5.3.1 Performance and Variation Model

The result of the multi objective optimization is a plot of objective space as shown in figure 5.4 and 5.5 for symmetrical and Miller-OTA respectively. The thick grey line on both of the plots are the Pareto-front of the objective space that represents the best optimal solutions for the design.

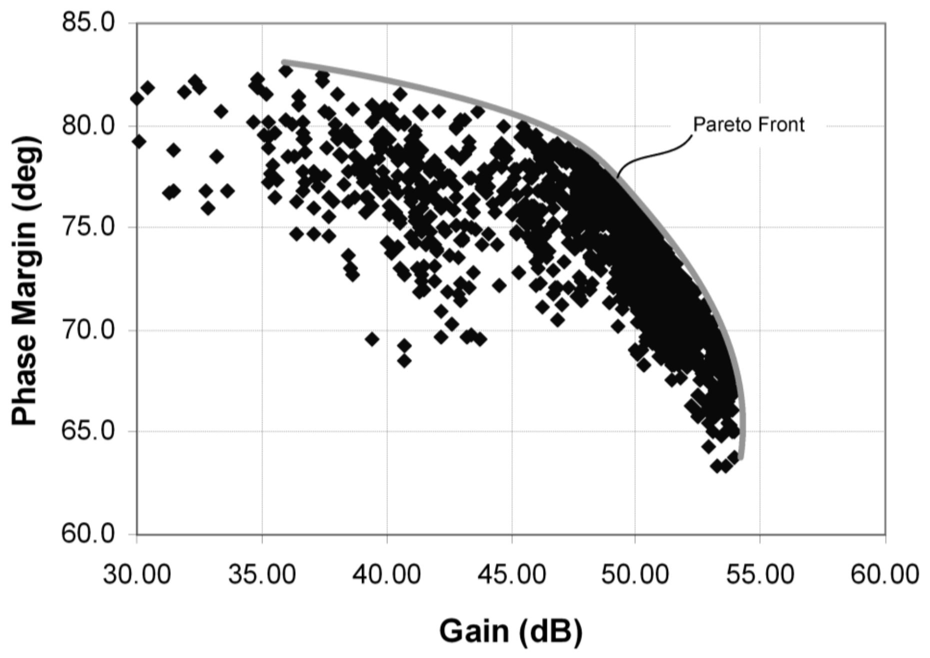


Figure 5.4: Symmetrical-OTA Pareto plot

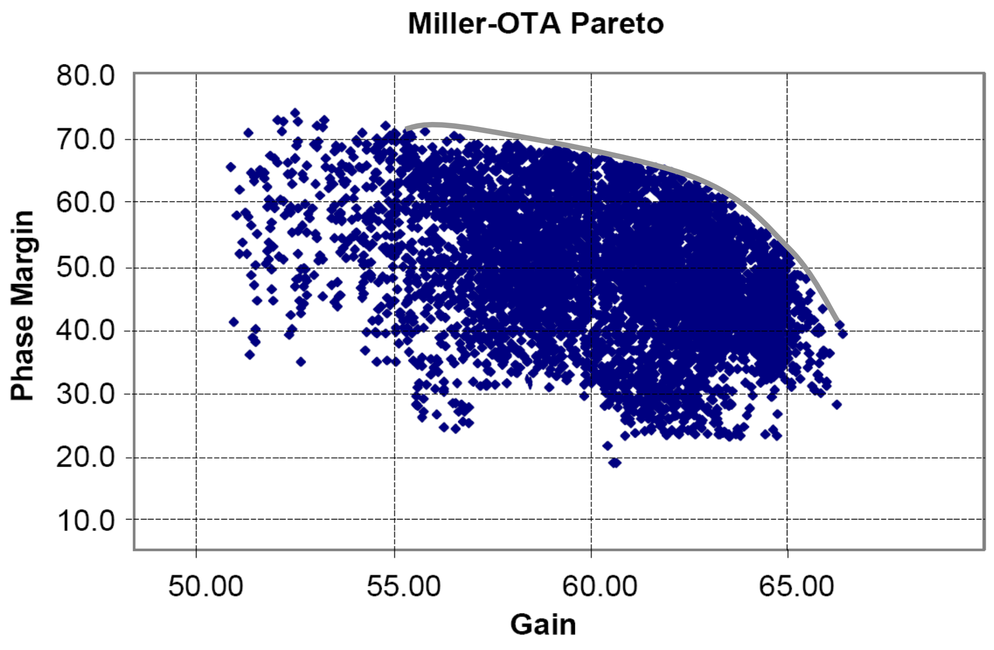


Figure 5-5: Miller-OTA Pareto plot

Once the Pareto-front of the design is determined, all the solutions on these curves are taken together with their corresponding design parameters. These information are stored in a text file which define the performance model for each topology.

The next step is to create the variation model for the Pareto-points. Every optimal solution on the Pareto-front undergoes a Monte Carlo simulation using process variation and mismatch models. 200 samples were chosen for the MC simulation and from these a standard deviation is calculated for each of the performances. The standard deviation values are multiplied by 6 for its 6th-standard deviation minimum and maximum variation. All the variations data for each of the Pareto-points are stored in another text file and represents the variation model for the circuit.

5.3.2 Table Model function implementation

The performance and variation behavior for the symmetrical OTA is modelled as a lookup table using a Verilog-A table model function. There will be two different table models that represent the performance behavior and the variation behavior for each of the performance point on the Pareto front. Table 5.2 shows some selection points of the Pareto front obtained from the multi-objective optimization.

Design:	**Gain (dB):**	**ΔGain (%):**	**PM (deg):**	**ΔPM (%):**
21	49.78	0.52	76.3	1.50
22	49.90	0.52	76.1	1.51
24	49.98	0.51	76.0	1.51
25	50.17	0.51	75.8	1.52
26	50.35	0.50	75.5	1.56
27	50.45	0.49	75.3	1.57
32	51.06	0.44	74.1	1.69
35	51.14	0.51	74.0	1.71
37	51.24	0.42	73.8	1.69
38	51.62	0.42	73.2	1.68

Table 5.2: Performance and Variation table

Figure 5-6 shows the table model data file for the OTA performance obtained from the Pareto front. For a given performance value (in this example, gain) the other feasible performance value can be interpolated by the table model function. The table model function for the performance model can be written as shown in equation 5-6.

pm = $table_model(gain, “pareto.tbl”, “3E”); 5-6

This statement will interpolate the phase margin performance from the given gain value. “pareto.tbl” is the file name and “3E” represents the interpolation and extrapolation type where cubic interpolation (‘3’) and no extrapolation (‘E’) are used. With the table model function, the feasibility of the performance can be maintained where the interpolation will only consider the values within the sampled domain. The variation table model can be used to determine the variation for each of the performances as shown by the data file in figure 5-7. The table model function for each of the performance variation can be written as shown in equation 5-7 and 5-8.

gain_var = $table_model (gain, “gain_var.tbl”, “3E”); 5-7

pm_var = $table_model (pm, “pm_var.tbl”, “3E”); 5-8

Based on the variation table (figure 5-7), the variation for a particular performance value can be interpolated. This interpolation will tell the minimum and maximum limit of the performance and can be used to determine how good the performance compared with the specification boundary and hence can be used to look for another solution that can maximize the yield. The resulting Verilog-A listing for the behavioral model is shown in figure 5-8.

```
# pareto.tbl
# table model example for
# symmetrical-OTA Pareto
front
# Gain  PM
49.78   76.3
49.90   76.1
49.98   76.0
50.17   75.8
50.35   75.5
50.45   75.3
51.06   74.1
51.14   74.0
51.24   73.8
51.62   73.2
```

Figure 5-6: Table model file for OTA performance model

```
# gain_var.tbl
# table model example for
#  gain variation of the Pareto front
# Gain  Variation(%)
49.78   0.52
49.90   0.52
49.98   0.51
50.17   0.51
50.35   0.50
50.45   0.49
51.06   0.44
51.14   0.51
51.24   0.42
51.62   0.42
```

```
# pm_var.tbl
# table model example for
# PM variation of the Pareto front
# PM    Variation(%)
76.3    1.50
76.1    1.51
76.0    1.51
75.8    1.52
75.5    1.56
75.3    1.57
74.1    1.69
74.0    1.71
73.8    1.69
73.2    1.68
```

Figure 5-7: Table model file for a)gain and b)phase margin variation model

```
analogue begin

   pm = $table_model(gain, “pareto.tbl”, “3E”);
   gain_var = $table_model (gain, “gain_var.tbl”, “3E”);
   pm_var = $table_model (pm, “pm_var.tbl”, “3E”);
   gain_new = ((gain_var)/100)*gain) + gain;
   pm_new = $table_model(gain_new, “pareto.tbl”, “3E”);
   $display (“Propose new gain value : %e” , gain_new);
   gain_in_v = pow(10, gain_new/20);
   V(out) <+ V(inp) * (-gain_in_v) – I(out) * ro;

end
```

Figure 5-8: Verilog-A model for OTA performance and variation lookup table

5.3.3 Interpolation example

The performance and variation model can be used to find a circuit solution for a given performance specification. This avoids the need to re-run the simulation-based optimization and will significantly reduce the design cycle time. To find a solution, the variation model will be used to interpolate a new performance value from a given specification. From the new performance value, a set of design parameters will be interpolated using the performance model. Table 5.3 shows an example for the interpolation where the required performance is a gain greater than 50dB and a phase margin of greater than 74 degrees.

The variation for gain and phase margin performance is obtained by interpolation from the table model function. In this case, the relevant look-up table points are those shown in Table 5.2 where it can be seen that the gain of 50dB is between design point 24 and 25. The variation interpolation given between these points is 0.51%. Using this variation value, it can be said that the actual gain may vary from 49.75dB to 50.26dB and therefore, in order to achieve maximum yield, the specified gain of the design must be at least 50.26dB. If we choose a design point with a 50.26 dB gain value, and with 0.51% variation, the gain will vary between 50.01dB to 50.51dB. This will ensure that the required 50dB gain will be achieved within the process extremes. The value of 50.26 dB therefore becomes the new targeted performance value and this value will be used to interpolate the feasible phase margin performance from the lookup table. From the lookup table (table 5.2), the phase margin value that will be interpolated based on 50.26dB gain is between 75.5 and 75.8 degrees. This value met the specification for the phase margin. The variation model of the phase margin is used to determine the variation of this new phase margin value. The interpolated variation is 1.53% which will make the phase margin to vary between 74.36 to 76.64 degrees. This variation is still within the given specification. With the new performance values for gain and phase margin, the design parameters that will give the required performances can be determined from the Pareto front.

Performance:	Required Performance:	Variation:	New Performance:
Gain	> 50dB	0.51%	50.26dB
Phase Margin	> 74 deg	1.53%	75.60 deg

Table 5.3: Interpolation example

5.3.4 Model Verification

To verify the performance and yield interpolated by the behavioural model, a comparison has been made with transistor level simulation using the design parameters obtained from the table model function. This comparison is shown in table 5.4. The percentage error in passband gain and phase margin was calculated between the OTA transistor simulation and interpolated values from the Verilog-A model. The error is the different between the transistor model and the behavioural model performance. Figure 5.9 shows the open loop gain for the Verilog-A model and transistor model. It can be seen from these comparisons that the Verilog-A function matches closely with the transistor level simulation. A Monte Carlo simulation using 500 samples was carried out and verified overall a yield of 100% for the OTA design.

Figure 5.9 shows a divergence in the comparison above 40MHz which is attributed to parasitic poles in the transistor circuit. Although these higher order effects are not modeled in this example, they could be incorporated if required. For example, figure 5.10 shows another example of the open loop gain comparison for Miller-OTA that includes the higher order effects that comes from parasitics poles in the circuit. A detail behavioural modelling of the OTA with all the parasitic poles will be discussed in chapter 6.

Performance Functions	Transistor Model	Verilog-A Model	% error
Gain	50.73	50.26	0.93%
Phase Margin	76.06	75.60	0.60%

Table 5.4: Performance comparison

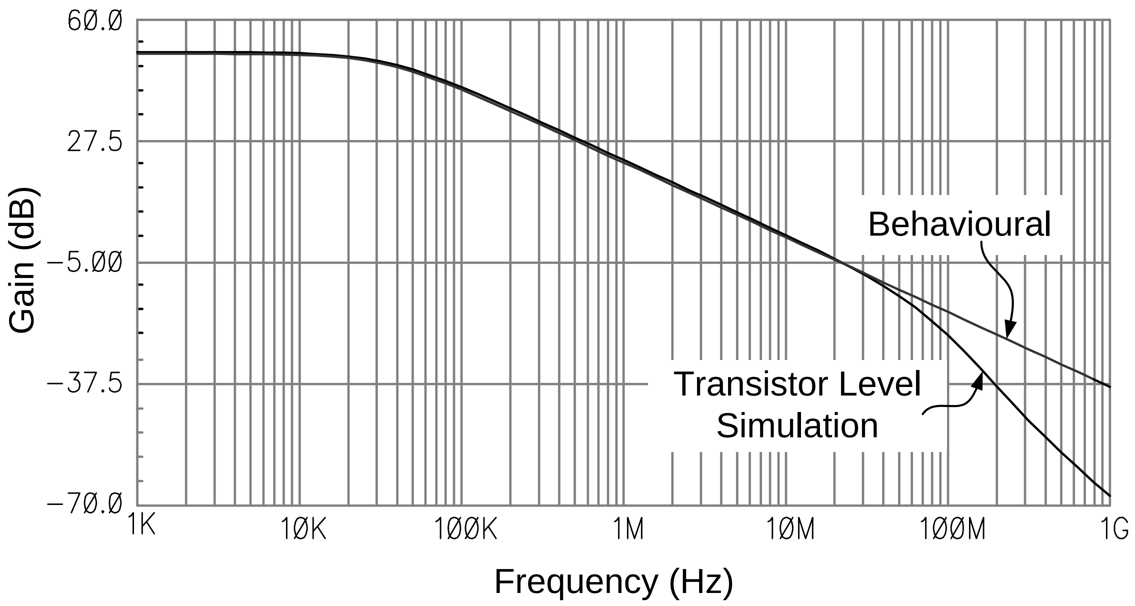


Figure 5-9: Behavioural and transistor level simulation comparison

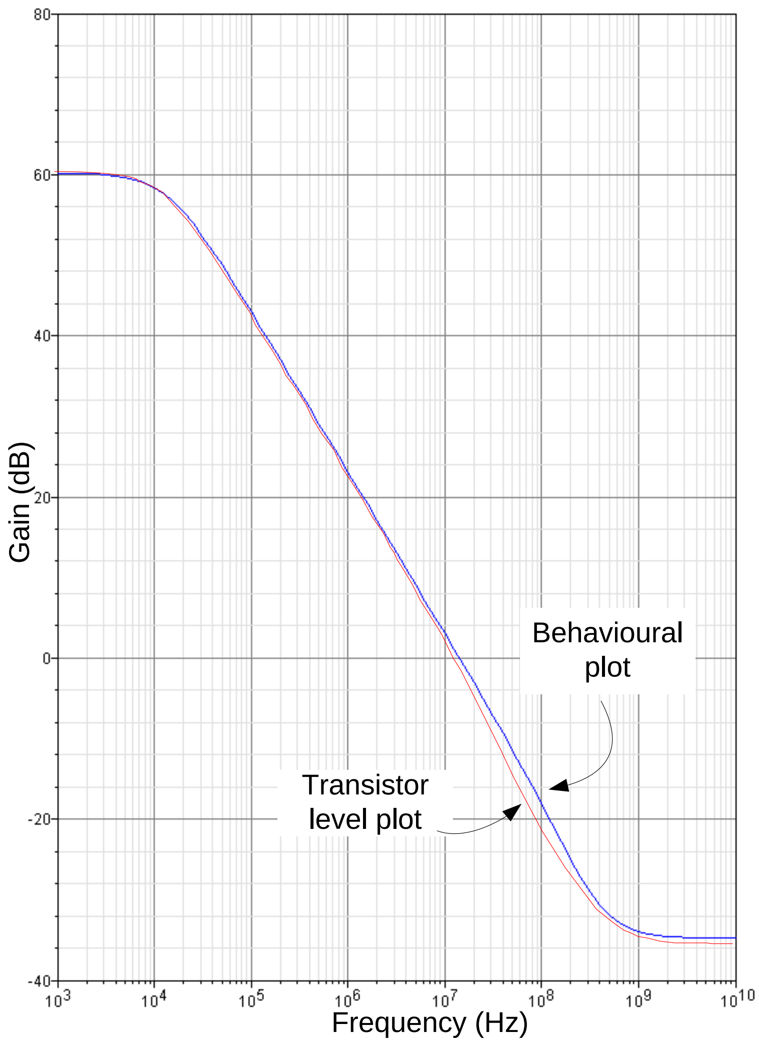


Figure 5-10: Open loop gain comparison for Miller-OTA

5.3.5 Topology Comparison

The interpolation example shown previously demonstrates how the table model function can be used to search a design solution for a particular circuit topology. However, the model will not find a solution if the new targeted performance is not feasible within the chosen topology. In this case, a search across a different topology could yield the solution. Figure 5-11 shows two Pareto-fronts for the symmetrical OTA and the Miller OTA. The Pareto-fornt can be used to search for a feasible solution. For example, assume the gain specification is >54dB and Phase Margin is >70 degrees. Looking at figure 5-11, these requirements are not feasible for symmetrical OTA but feasible for Miller OTA as shown by the shaded area. Therefore, in this case the performance and variation model of Miller OTA must be used to interpolate the variations and to find the design solutions for the requirements. This come in handy if a library of Pareto-front and the performance and variation model can be developed for a various type of circuit topology.

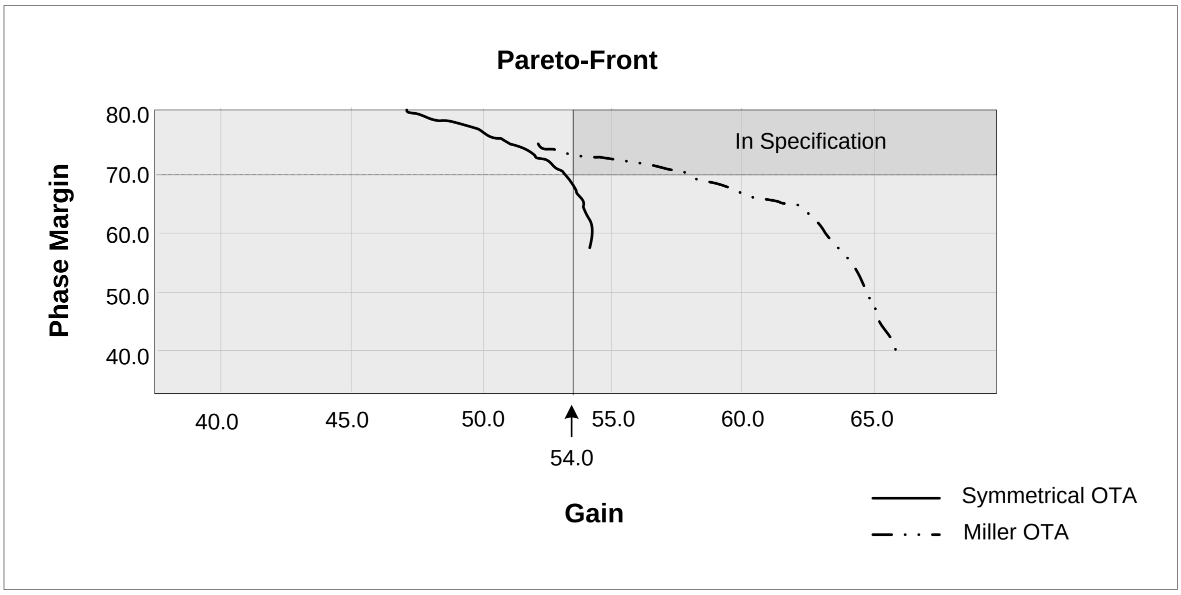


Figure 5-11: Pareto comparison between topology

5.3.6 Summary of Examples

Table 5-5 summarizes the model development activity. A total of 10,000 simulations were run in the initial MOO step for the performance model for both of the OTA topologies and Monte Carlo analysis was performed on 1022 Pareto Optimal points of symmetrical OTA and 987 points of Miller-OTA for the variation model. The whole model development stage took 4 hours to complete for the symmetrical OTA and 3 hours 40 minutes for the Miller-OTA on a 1.2GHz Ultra Sparc 3 computer system.

The effort involved in developing the performance and variation model can be compared with the transistor level optimization strategy such as that used in NeoCircuit optimization. Refer back to NeoCircuit optimization example for symmetrical OTA shown in chapter 4, which requires 1hr 29 minutes to optimize the OTA, the cost involved for the symmetrical OTA model development (in terms of CPU time) therefore will be paid off after 3 repeated uses.

Parameters:	**Symmet-OTA:**	**Miller-OTA:**
No. Generations	100	100
Evaluation Samples	10,000	10,000
Pareto Points	1022	987
CPU Time (1.2GHz Sparc 3)	4 hours	3h 40m

Table 5-5: Summary of examples

5.4 Application Example

5.4.1 System level design

The combined performance and variation model developed in previous example was used to design a 2nd order low pass filter. The filter topology is shown in figure 5-12 and was designed to the anti-aliasing specifications shown in figure 5-13. The specifications for the open loop gain and phase margin for the OTA are 60dB and 60 degrees respectively. Based on the OTA specifications, a feasible topology is selected.

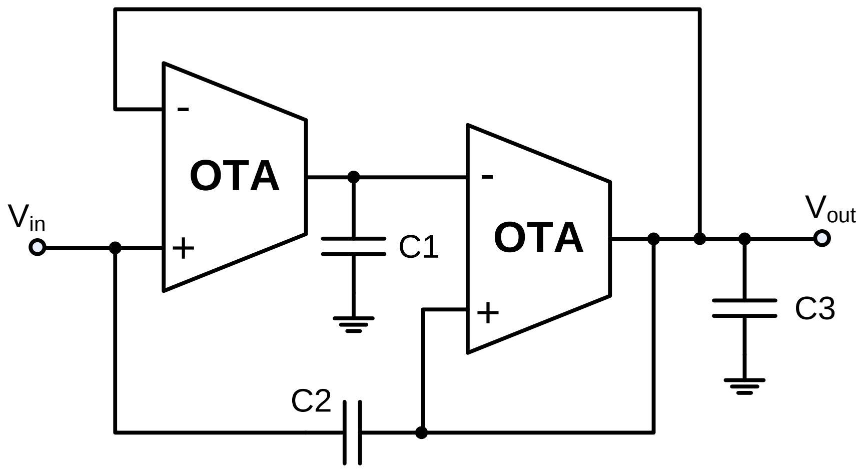


Figure 5-12: 2nd order lowpass filter topology

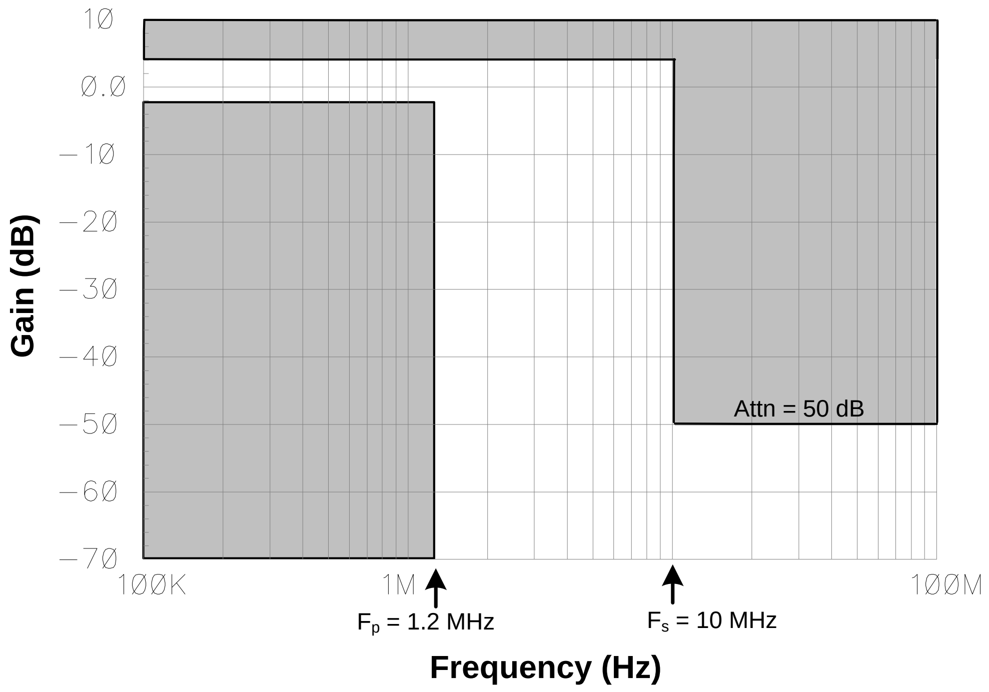


Figure 5-13: Filter specification

As explained in previous example, the Pareto plot can be used to compare topologies in order to choose which is feasible. In this case the Miller-OTA topology satisfies the specifications and was selected for the filter design. The performance and variation model of this OTA was used to select the OTA solution that met the specifications taking into account their variations. Table 5-6 shows some selection samples of the Miller-OTA design points with their performance and variation values.

Design:	Gain (dB):	ΔGain (%)	PM(deg)	ΔPM(%)
45	59.98	0.52	68.0	1.50
46	60.17	0.62	66.8	1.51
47	60.35	0.61	66.1	1.51
48	60.45	0.61	65.3	1.52
49	61.06	0.60	65.0	1.51
50	61.24	0.59	64.2	1.52
51	62.48	0.61	60.9	1.53
52	62.71	0.61	59.1	1.53

Table 5-6: Miller-OTA performance and variation values

From the table, 3 design points (48 ~ 50) meet the OTA specifications when variation is considered. Design point 47 fail the gain performance due to the variation and design point 51 fail the phase margin performance due to the variation. The chosen design points (that meet the OTA specifications) are then used in another multi-objective optimization for the filter in order to find an optimum solution for capacitor values C1, C2 and C3. Table 5-7 shows the result of this optimization. Monte Carlo analysis was performed on all the design solutions to find the solution with the highest yield.

Design Points:				Performance:			
OTA	C1	C2	C3	Attn	fp	fs	Yield
OTA1	575.5	2.412	759.2	57.85	1.21	8.56	54
OTA1	612.1	2.171	695.6	53.21	1.59	9.11	100
OTA2	564.0	2.160	817.8	54.65	1.55	8.17	100
OTA2	542.5	2.540	951.2	55.21	1.42	7.69	95
OTA2	480.1	2.493	854.7	59.21	1.18	8.62	27
OTA3	521.2	2.566	766.2	50.12	1.65	9.76	67

Table 5-7: 2nd order low pass filter optimisation results

For verification, a circuit level simulation of the sized low pass filter is used. A Monte Carlo simulation with 500 samples confirmed a yield of 100%.

5.4.2 Silicon Prototype

A silicon prototype for the 2nd order low pass filter designed previously based on the proposed methodology was developed and fabricated. Figure 5-14 shows a layout view of the designed chip. A test board for the chip measurement was designed as shown in figure 5-15. The chip performance has been measured and compared with simulation data.

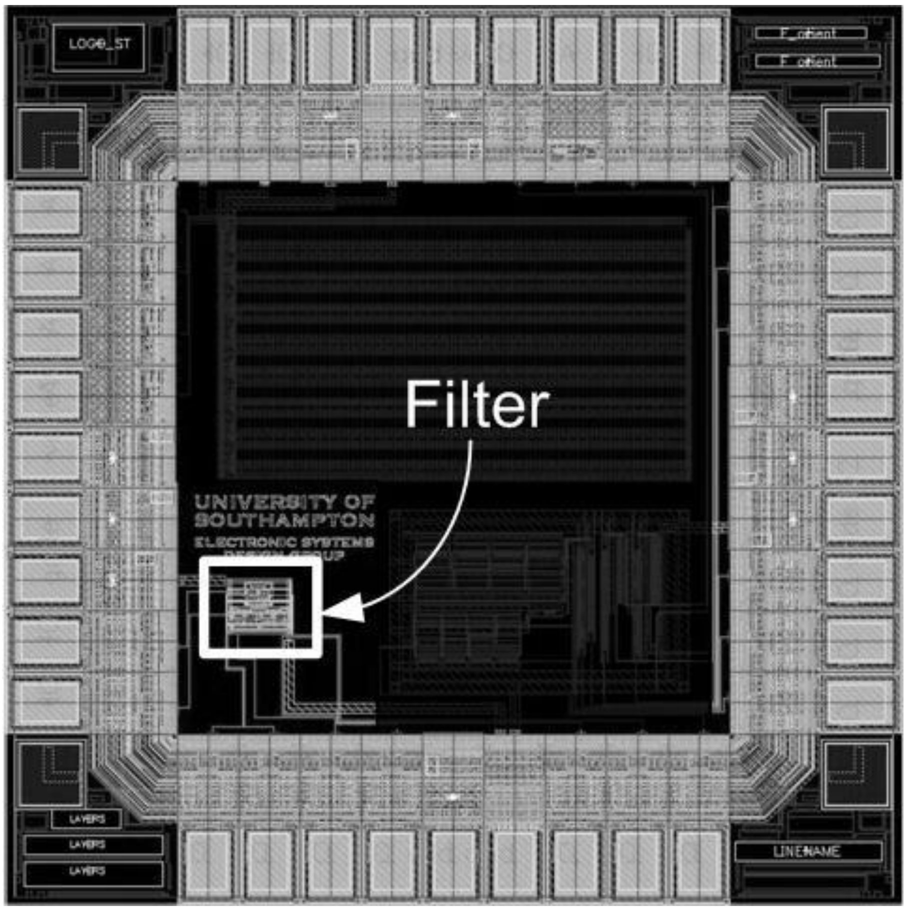


Figure 5-14: Layout view of silicon prototype

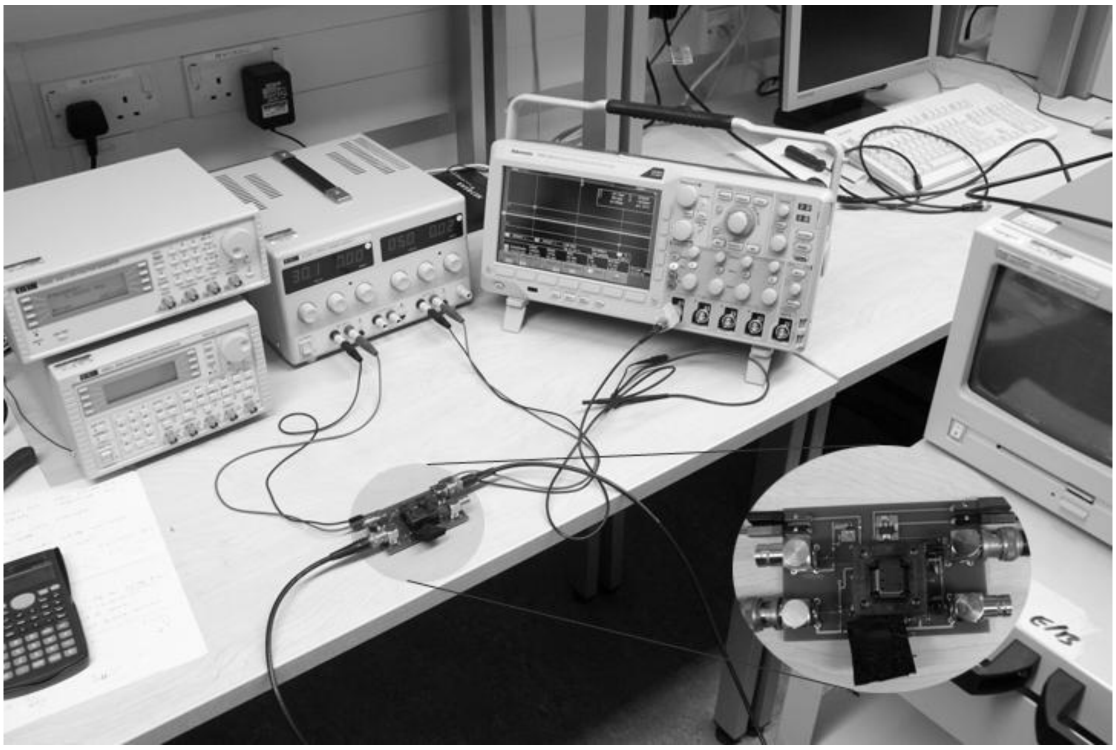


Figure 5-15: Test board snapshot

Figure 5-16 shows the filter response of all prototype samples overlaid with the simulation plot, showing that all the prototypes closely match within ±3% with the simulation data. These results confirm the accuracy and effectiveness of the technique in practice.

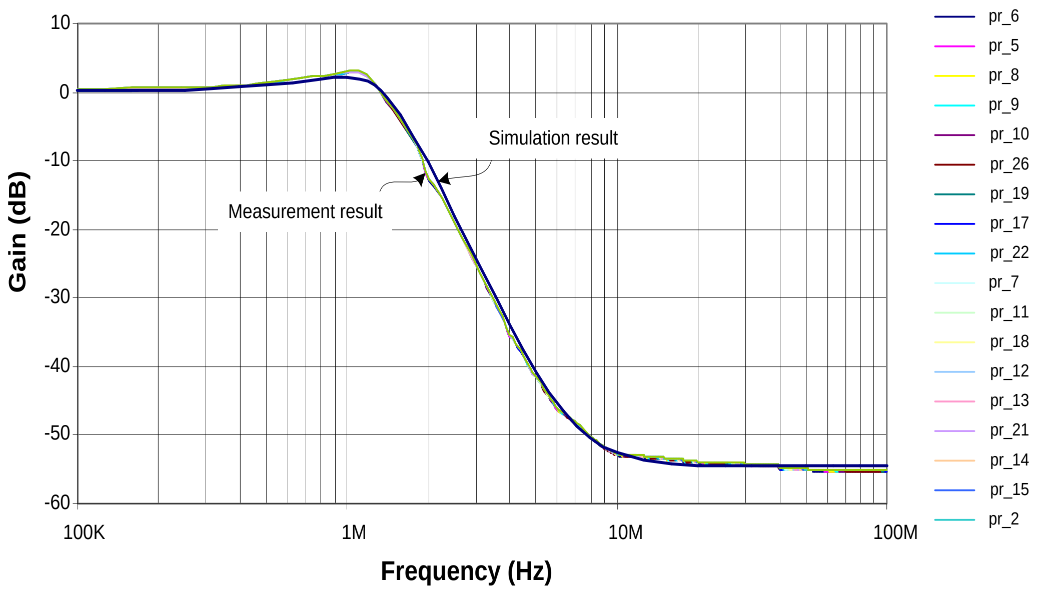


Figure 5-16: Chip measurement result

5.5 Summary

This chapter has presented a new approach that combines performance and variation objectives in a behavioral model for analogue circuits. Multi-objective optimization based on an evolutionary algorithm is used to explore tradeoffs between performance and yield, leading to a set of Pareto optimal solutions for the design. Monte Carlo variation analysis is performed on all the Pareto optimal solutions, and a table is constructed for both the performance and variation analysis. A behavioral model developed in Verilog-A is used together with this table to determine the parameters required to achieve the highest yield within a given specification. The model developed can be used in a hierarchical system design and demonstrates significant benefits especially in terms of design cycle time. After the initial time investment to create the model, there are significant improvements in overall simulation time and efficiency compared to conventional simulation based approaches. These benefits are enjoyed without a corresponding drop in accuracy. Two benchmark OTA topologies and a standard filter design have been presented to demonstrate the proposed algorithm and the behavior has been verified through transistor level simulations and measured silicon results.

Chapter 6

Hierarchical-based Design Optimisation

6.1 Introduction

A simulation-based design approach usually requires a high CPU computational effort as has been demonstrated in the examples from chapter 4 and 5. This is due to the fact that the performance of the circuit must be evaluated for a large number of different circuit variables, a process known as design space exploration. The bigger the circuit, the bigger the design space that must be explored. Running the entire performance evaluation at transistor level is computationally intensive. Therefore most of the tools developed using this approach are limited to rather small building blocks [84, 85]. Due to the increasing complexity of electronic systems and high demand of design cycle time reduction, the research focus for large analogue mixed-signal system has shifted towards a hierarchically based design technique. Hierarchical design employs divide and conquer approach involving breaking down a large system into its smaller constituent building blocks that can be designed and optimized individually.

There are many methodologies available for designing a large system depending on how the performance and design space are organized and traversed [86]. The design space of a complete system can be handled as a whole where all design variables in the system are considered at once (known as `flat' design) or it can be organized hierarchically into sub-systems and traversed according to the hierarchical flow. There are many methods available for hierarchical design which will be discussed in the remainder of this chapter.

The discussion starts with a brief overview of standard hierarchical design methods. After outlining the basic structure of hierarchical design flow, a new methodology that combines the Pareto modelling and top-down design of a system is proposed. As a design application, a 7th order elliptic low pass filter is used to demonstrate the proposed methodology. This example will demonstrate a complete design flow from bottom-up performance and variation modelling for the sub-block circuit and top down design for the whole system.

6.2 Hierarchical-based design

Generally, a hierarchical design methodology consists of a top-down design and bottom-up verification process as depicted in figure 6-1 [87],[59]. The whole process is based on two important design aspects: circuit decomposition and specification propagation. Circuit decomposition involves breaking down the system level architecture into smaller, less complex, subsystems. When the subsystems are still too complex to design, a second decomposition is performed. This decomposition will continue until all subblocks are manageable for design. The lowest hierarchical level is the transistor level where the block can be simulated by a Spice-like simulator to extract its performance. Specification propagation involves translation of system level specifications into lower level specifications. This is a very important aspect of hierarchical design in order to avoid the failure to find optimum design due to non-feasible solutions that may occur if the lower level blocks can not meet the system level specifications. On top of that, the specification propagation step helps to determine the system level yield. The yield is defined based on the system level specifications but it is determined by the circuit level variations. Therefore, in order to predict and optimise the yield for the system level design, the variation of the lower level must be propagated to the top level.

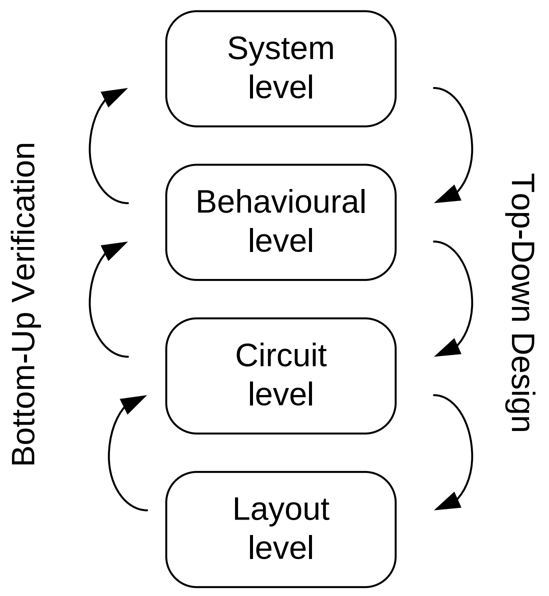


Figure 6-1: Hierarchy design methodology

6.3 Hierarchical-based Design Methodology

Several hierarchical-based methodologies exist that can be used to overcome the design complexity of a large mixed-mode system. This section discuss some of these methodologies.

6.3.1 Bottom-Up Methodology

In this method, the design starts with the system specifications. Based on the designer's knowledge, the system is broken down into sub systems until reaching the lowest level of transistor blocks. Next, all the blocks are designed in a bottom-up fashion. To cope with the feasibility problem, the lower-level blocks tend to be overdesigned. Once the design reaches the top level, the design performances are checked and compared with the specifications. If the system fail to meet the specifications, a complete bottom-up redesign may need to be done all over again which consumes precious design time.

6.3.2 Top-down Constraint-Driven Methodology (TDCD)

TDCD methods traverse the design hierarchy, starting from a set of system level specifications. Starting from the system level specifications, an architecture is chosen, and designed (optimised) at the architecture level using an optimiser. In [88, 87, 89], a set of equations were used to describe the feasible performance and the optimisation at the architecture level was done towards the objective to maximise the flexibility. The design space at this level is the objective space for the next lower-level block. In this way, each sub-block will have their own specifications to be met. This is how the system level specifications are propagated or transferred to the lower level. The next lower level is then optimized in the similar way and the hierarchy is traversed down until transistor level. During the transformation, if the sub-blocks are not feasible or specifications cannot be met, the hierarchy is climbed-up again and a new architecture is selected. Once all the hierarchy levels have been designed and the transistor level block has been sized accordingly, a full bottom-up verification will be performed with accurate transistor level simulation. In [90], the TDCD method was used as a part of the simulation-based synthesis tool for analogue cell sizing called AMIGO. Here, the subblock level performance parameters were used as the design variables for the system level optimisation. Thus the performance of the lower level is specified while optimising the system level block. Later, the lower level block can be optimised separately to determine the transistor level parameters.

6.3.3 Feasibility Modelling Bottom-up (FMBU) + TDCD

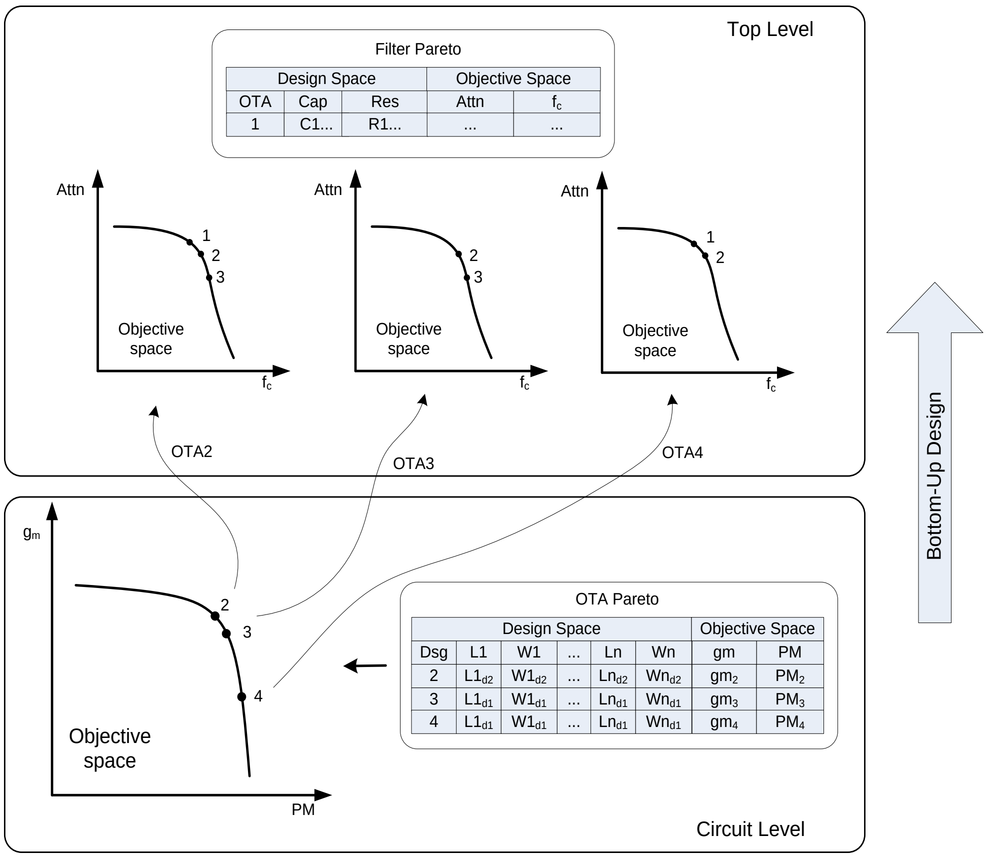
The TDCD approach discussed previously suffers from feasibility problems and the need to climb-up the hierarchy level several times if it fails to find feasible sub-blocks. Due to this limitation, researchers have focused on developing the feasibility model of a performance space in a bottom-up fashion and then followed by a TDCD flow. The radial basis function [91], support vector machine [92] and spec-wise linearised models [93] have been used to model the feasibility and the performance space of the sub-block level. With this model, it can be repeatedly used without the need to re-run the optimisation and over time, libraries of feasibility models can be built. The main disadvantage of this methodology is considerable simulation effort is expended to model the whole feasibility region which includes all the design points, including optimum and non-optimum design points.

6.3.4 Multi Objective Bottom-up Methodology (MUBU)

MUBU approach consists of two important ideas :

- only consider performance's trade offs rather than the whole objective space and
- to use designed circuits rather than models.

The development in analogue CAD leads to the concept of multi objective optimisation and Pareto-points which has been explained earlier in this thesis. In MUBU method, the circuit/cell level Pareto points are directly exploited for system level design. The design space of next level up is the selection of design space for each of the sub-blocks and the hierarchy traversal proceeds in an upward flow as illustrated in figure 6-2. This idea has been used in chapter 5 of this thesis for the example of 2nd order low pass filter design.



Filter Pareto

Design Space			Objective Space	
OTA	Cap	Res	Attn	f_c
1	C1...	R1...	...	...

OTA Pareto

Design Space					Objective Space	
Dsg	L1	W1	...	Ln	Wn	gm	PM
2	$L1_{d2}$	$W1_{d2}$	...	Ln_{d2}	Wn_{d2}	gm_2	PM_2
3	$L1_{d1}$	$W1_{d1}$	...	Ln_{d1}	Wn_{d1}	gm_3	PM_3
4	$L1_{d1}$	$W1_{d1}$	...	Ln_{d1}	Wn_{d1}	gm_4	PM_4

Figure 6-2: Multi Objective Bottom Up hierarchical methodology

In contrast with TDCD, any design selected on any level in MUBU method is already fully sized. Once the designer selects a solution at the system level that meets the specifications, the design variables of the complete system have been specified. In this approach, there is no need for specification propagation since all the optimum performance trade-offs are being used at the system level and at system level, the one that meet the specifications is chosen as the design solution. The Pareto optimal set generated can be reused and can compensate the cost involve during the optimisation process. Compared with the FMBU+TDCD method, which is applied to the whole performance space, MUBU only consider the performance trade-offs and only captures the good circuit candidates for the sub-block circuits.

6.4 Multi Objective Bottom Up (MUBU) + TDCD Architecture

In the MUBU approach, the design space for the next level up is the selection of a design for each of the sub-blocks. However, in most cases, once the system level specifications have been specified, it does not specify the requirements for the lower level blocks. Therefore in the MUBU approach, to optimise at the system level, the algorithm needs to jump among discrete points of the Pareto in order to find the solution that meet the system level requirements. If all the solutions from the sub-blocks level do not meet the system level specifications, the sub-blocks topology is not feasible for the design and a new Pareto-points for a different topology need to be created. Then, the system level optimisation need to be repeated again to find the solutions.

Most of the hierarchical-based methodologies discussed so far do not consider performance variation in the design flow hence they are unable to predict and optimise the system level yield. In order to optimise the yield at system level, it is necessary to take into account the variation of the sub-blocks level and there must be a way to exploit this information during the bottom-up flow. This chapter proposes a new hierarchical-based design methodology that model the variation of the sub-blocks performances that can be used for system level yield prediction. The methodology is illustrated in figure 6-3.

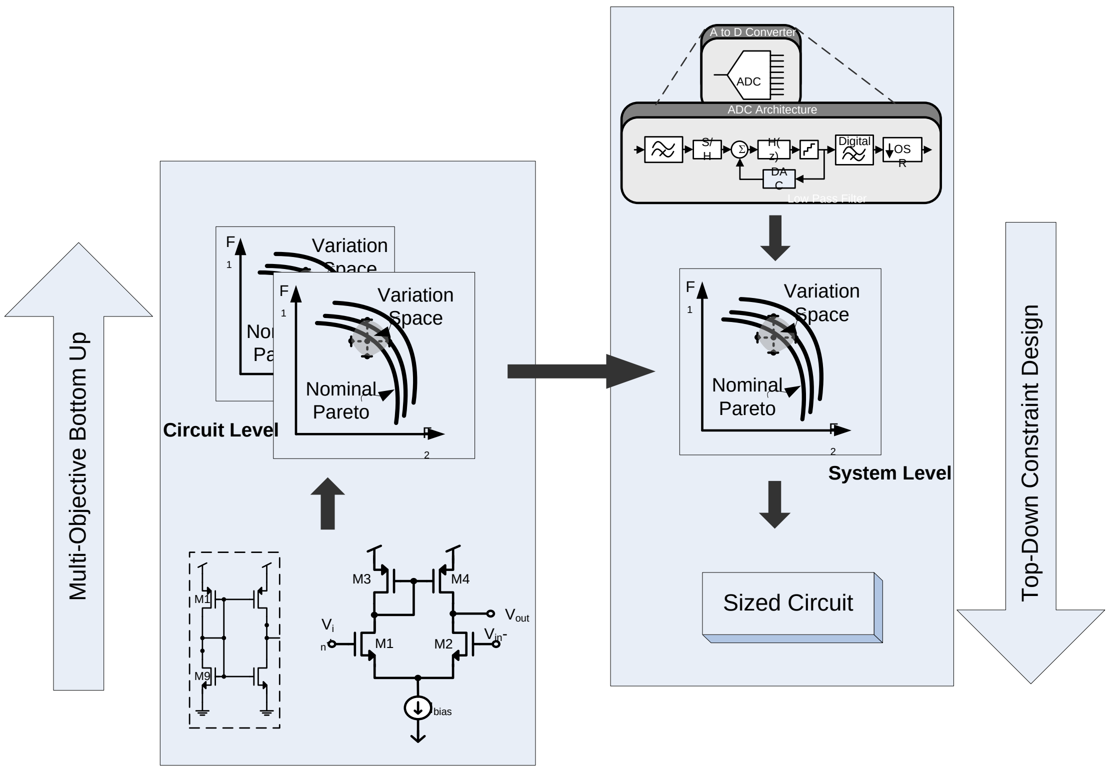


Figure 6-3: MUBU + TDCD Architecture

In this proposed approach, the multi objective bottom up design flow is used to develop a performance and variation model of a sub-block circuits. Pareto points from a multi objective optimisation will be extracted for the performances and Monte Carlo simulation is applied on the Pareto-points for the variation modelling. Standard deviation of the Monte Carlo result is calculated and a 6-sigma minimum and maximum range is estimated. Both the performance and variation are modelled in a lookup table using behavioural language which later can be used for system level design and optimisation. Once the model has been developed, a TDCD method is applied for the system level design. At the system level, behavioural modelling is used for the optimisation and the system is optimised towards the system specifications. With the inclusion of variation model from the sub-block level, the performance space of the system level will include their performance variations. As a result, a solution that meets the specification for nominal performances and its tolerances can be selected which in turn will maximise the overall yield. The design parameters of the system level will be the target specifications for the next lower-level sub-blocks. The lower-level performance and variation model will be used to select the design parameters that meet the lower-level specifications. This top-down design process flow will continue until the hierarchy reaches the transistor level. At the transistor level, the whole system design has been sized to meet the system level specifications and at the same time produce higher yield.

6.5 Design Example: 7th order elliptic low pass filter

To demonstrate the proposed methodology, a system level of 7th order elliptic low pass filter is used as a design example. This section presents a complete design flow starting from behavioural performance and variation model development for single stage operational transconductance amplifier (OTA) to top down design strategy for the whole filter system.

6.5.1 Circuit Decomposition

In a hierarchical-based design the flow starts with breaking down the system level design into sub-blocks which is known as circuit decomposition. Therefore from a system level description, the architecture/topology of the system and each of the hierarchy level have to be determined until it reaches to the lowest transistor level. Figure 6-4 shows the break down of the 7th order elliptic low pass filter system.

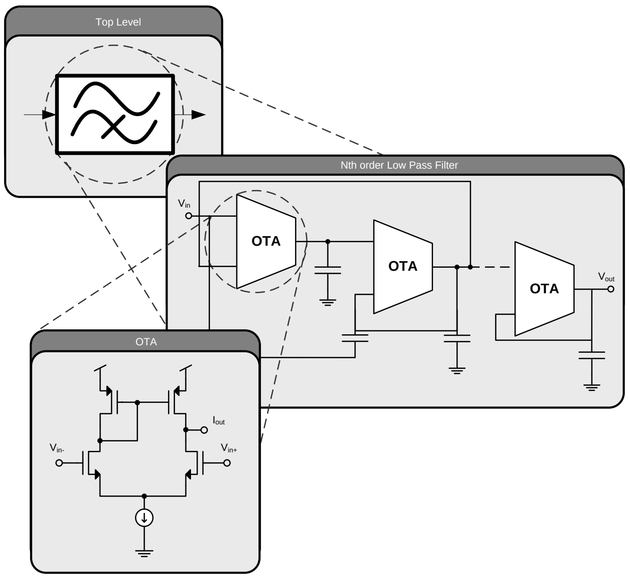


Figure 6-4: Circuit Decomposition

6.5.2 MUBU modelling – Design Initialisation

Multi objective bottom up methodology is used to develop the performance and variation model for the sub-block circuit. In this case, the model that will be developed is for the single stage OTA as shown in figure 6-5.

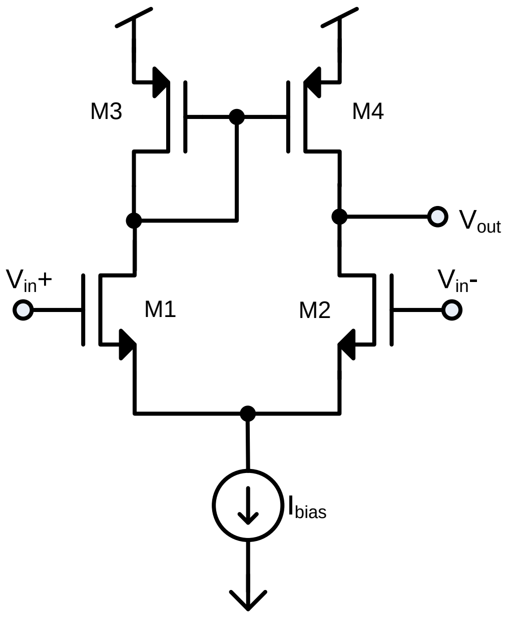


Figure 6-5: Single stage OTA topology

The first step in the model development is to determine the designable parameters for the topology. In this example, these are the transistor lengths and widths which make up a total of 4 designable parameters. In order to avoid mismatch in the design process for input pair and current mirror pair, the transistors are grouped as pair so that transistor M1 and M2 will have the same length and width and so does current mirror pair (M3 & M4). Three objective functions have been chosen for this example: transconductance (gm), output resistance (ro) and phase margin (pm). For this example, only three objective functions are chosen which is necessary and sufficient for the system level design in order to reduce the number of simulations needed for the multi objective optimisation. However, the performance objective is not limited to any number and it can be as many as required if a generic OTA model is to be developed and can be used in wide number of applications. Once the objectives have been defined, a spice netlist including the testbench for each of the performance objective is created. In this example, only one testbench is required as all the performance objectives can be simulated using single testbench with ac analysis.

6.5.3 MUBU modelling – Optimisation

Once the designable parameters have been determined, a GA string can be constructed as shown in figure 6-6. As explained earlier in this thesis, the multi objective optimisation has constrained some parameters including the decision space range. The algorithm chosen for the multi objective optimisation is Non-dominated Sorting Genetic Algorithm - II (NSGA-II) [51]. A brief overview of the NSGA-II algorithm has been presented in chapter 2 and the code for the algorithm is shown in Appendix B. The NSGA algorithm will generate the designable parameters according to the GA string and the range constraints. These parameters are used in the spice netlist for the performance evaluations. A total of 50 generations each with a population size of 400 were used in this case, giving 20,000 total samples for the optimisation.

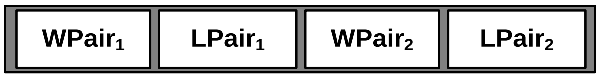


Figure 6-6: GA string for the design example

The testbench netlist is used to evaluate the performance for each design parameter set (defined by GA) and the result of the simulations determines the fitness score of the individuals. A non-dominated sorting and crowding distance sorting are applied to the solution for each generation in order to find the final diverse set of Pareto-fronts. The result of the optimisation is a full set of designable parameters confined by the parameters range and their corresponding performance functions.

6.5.4 MUBU Modelling – Performance and Variation Model

The outcome of the previous multi objective optimisation for the OTA is a set of optimal solution called Pareto-front. The Pareto points are the best performance trade-offs among the competing objectives for the circuit. All the solutions on the Pareto front are taken as the optimal performances and will be defined as the performance model for the OTA. The variation model for the Pareto points is developed with a

Monte Carlo simulation using process variation and mismatch models given by the foundry. 200 samples were chosen for the Monte Carlo simulation and from these the standard deviation of the sample is calculated. The standard deviation is multiplied by 6 for the 6-sigma minimum and maximum range. The minimum and maximum data represent the variation model. This together with the performance data is stored in a data file. As explained in chapter 5, a lookup table is used to model the performance and variation of the circuit. The look-up table is defined using Verilog-A behavioural language with $table_model() function as given in figure 6-7. Table 6-1 shows a selection of the lookup table sample points that include the performance functions and their variations.

```
analogue begin
 gm_delta = $table_model (gain, "gm_delta.tbl", "3E");
 ro_delta = $table_model (ro, "pm_delta.tbl", "3E");
 pm_delta = $table_model (pm, "pm_delta.tbl", "3E");
 gm_prop = ((gm_delta/100)*gm)+gm;
 ro_prop = ((ro_delta/100)*ro)+ro;
 pm_prop = ((pm_delta/100)*pm)+pm;
 p1 = $table_model (gm_prop,ro_prop,pm_prop, "p1_data.tbl","3E,3E,3E");
 p2 = $table_model (gm_prop,ro_prop,pm_prop, "p2_data.tbl","3E,3E,3E");
 p3 = $table_model (gm_prop,ro_prop,pm_prop, "p3_data.tbl","3E,3E,3E");
 p4 = $table_model (gm_prop,ro_prop,pm_prop, "p4_data.tbl","3E,3E,3E");
 fptr=$fopen("params.dat");
 $fwrite(fptr, "\n Generated Design Parameters\n ");
 $fwrite(fptr, "%e %e %e %e", p1,p2,p3,p4);
 $fclose(fptr);
 $display ("params: = %e %e %e %e", p1, p2, p3, p4);
End
```

Figure 6-7: Verilog-A table model function

Design:	gm :	Δgm:	ro :	Δro:	pm:	Δpm :
2	109μ	0.75%	382k	0.75%	87.9	1.74%
3	109μ	0.75%	384k	0.75%	87.8	1.73%
19	110μ	0.74%	371k	0.74%	88.0	1.73%
34	111μ	0.75%	497k	0.74%	85.3	1.71%
35	111μ	0.73%	375k	0.75%	87.9	1.73%
61	112μ	0.73%	458k	0.74%	86.1	1.71%
209	120μ	0.70%	486k	0.74%	82.7	1.70%
211	120μ	0.70%	743k	0.72%	74.9	1.69%

Table 6-1 Performance and Variation Samples

6.5.5 TDCD flow – Behavioural Description

Once the multi objective bottom up model development has completed, a top-down constraint design (TDCD) can be started. This design flow starts with system level optimisation and transformation of the system level specifications to bottom level blocks. In order to run system level optimisation, a behavioural model is used to describe the system. This approach offers fast simulation and optimisation hence the optimum solutions for the system can be quickly determined. Therefore, a complete behavioural model has to be developed for the system level taking into account all the sub-block circuits. The behavioural performance and variation model developed during the MUBU stages can be combined together with the system level behavioural to find the solution.

In this example, a behavioural model for an OTA is developed based on ac small signal analysis. The OTA topology used in this example is not symmetrical hence both side of the differential pair (LHS & RHS) must be taken into consideration for the analysis. The differential input signal applied to the input is given by equation 6-1 and the input signal is given by equation 6-3.

$$v_{id} = v_{inp} - (-v_{inm}) \qquad 6\text{-}1$$

$$= 2vin \qquad 6\text{-}2$$

$$\therefore vin = v_{id} / 2 \qquad 6\text{-}3$$

Where v_{id} is the differential input signal, v_{inp} is the positive input and v_{inm} is the negative input. Each side of the differential pair will be analysed individually to derive the dc gain of the circuit. Figure 6-8 shows the small signal model for left hand side (LHS) and right hand side (RHS) of the OTA.

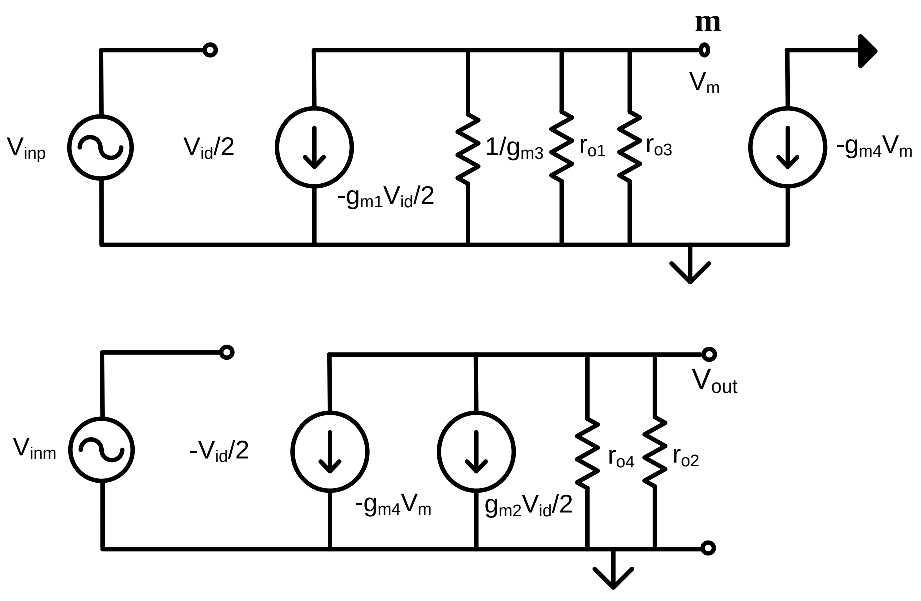


Figure 6-8: OTA small signal model

In the above model, the voltage at node m is given by equation 6-4 :-

$$v_m = -gm_1\left(\frac{v_{id}}{2}\right)\left(ro_1 // ro_3 //\left(\frac{1}{gm_3}\right)\right) \qquad 6\text{-}4$$

Since $\frac{1}{gm_3}$ is very small, equation 6-4 can be reduced to :-

$$v_m = -gm_1\left(\frac{v_{id}}{2}\right)\left(\frac{1}{gm_3}\right) \qquad 6\text{-}5$$

Which can be re-written as :-

$$v_m = -\frac{gm_1}{gm_3}\left(\frac{v_{id}}{2}\right) \qquad 6\text{-}6$$

At the RHS, the voltage at the output of the OTA is given by equation 6-7.

$$v_{out} = \left[gm_2\left(\frac{v_{id}}{2}\right) + \left(-gm_4 v_m\right)\right]\left(ro_4 // ro_2\right) \qquad 6\text{-}7$$

Apply equation 6-6 into 6-7 for v_m :-

$$v_{out} = \left[gm_2\frac{v_{id}}{2} + \frac{gm_4}{gm_3}gm_1\frac{v_{id}}{2}\right]\left(ro_4 // ro_2\right) \qquad 6\text{-}8$$

Since $gm_1 = gm_2 = gm_3 = gm_4 = gm$, equation 6-8 can be re-written as :-

$$v_{out} = gm.v_{id}\left(ro_4 \,//\, ro_5\right) \qquad 6\text{-}9$$

Therefore, the gain for the OTA, A_v is :-

$$A_v = \frac{v_{out}}{v_{id}} = gm\left(ro_4 \,//\, ro_2\right) \qquad 6\text{-}10$$

The above analysis represents the dc gain for the OTA at low frequency. To accurately analyse the behaviour of the OTA for high frequency operation, all parasitic capacitances have to be considered. Figure 6-9 and 6-10 show the OTA schematic with parasitic capacitance and its small signal model respectively. C_n in the small signal model is the total capacitance at the input node, n and C_o is the total capacitance at the output node, o.

$$C_n = C_{gs3} + C_{gs4} + C_{db1} + c_{db3} \qquad 6\text{-}11$$

$$C_o = C_{db2} + C_{gd4} + C_{db4} + c_L \qquad 6\text{-}12$$

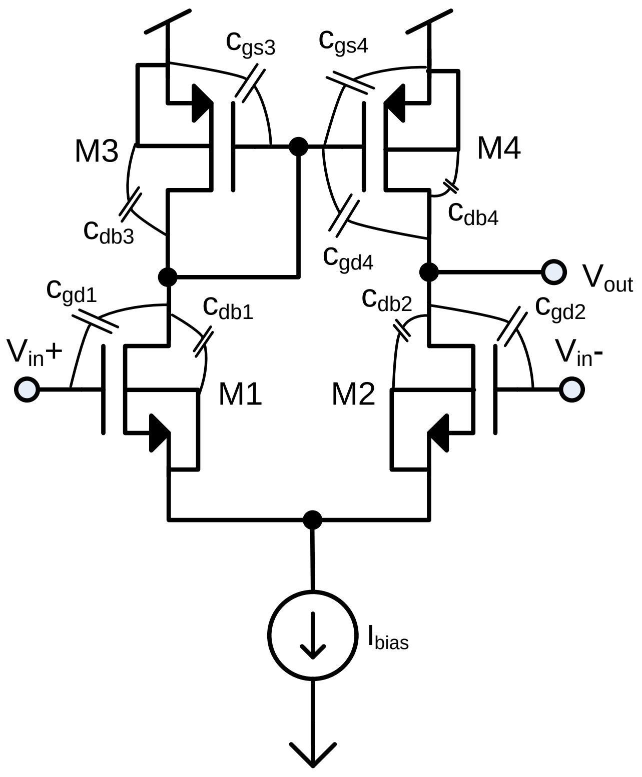
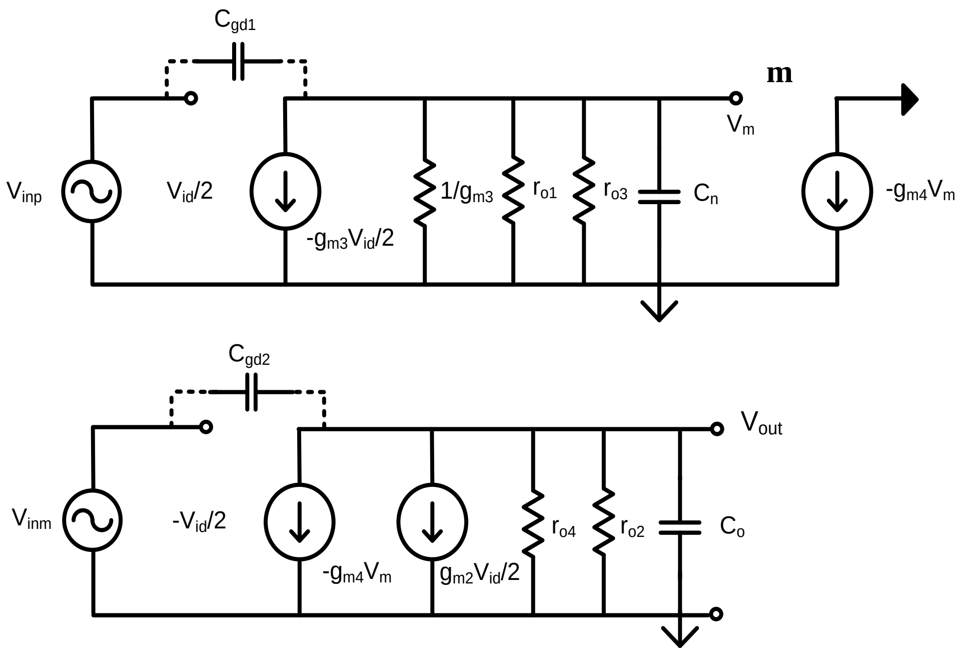


Figure 6-9: OTA schematic with parasitic capacitance



$$C_n = C_{gs3} + C_{gs4} + C_{db1} + C_{db3}$$

$$C_o = C_{db2} + C_{gd4} + C_{db4} + C_{CL}$$

Figure 6-10: OTA High frequency small signal model

At LHS, voltage at node m can be written as :-

$$v_m = -gm_1 \frac{v_{id}}{2}\left[\frac{1}{gm} // C_n\right] \qquad 6\text{-}13$$

Reactance obtained from $1/gm_3$ parallel with Cn can be expressed as :-

$$X_s = \frac{1}{gm_3 + sC_n} \qquad 6\text{-}14$$

Therefore, v_m can be written as :-

$$v_m = -\frac{gm_1 \frac{v_{id}}{2}}{gm_3 + sC_n} \qquad 6\text{-}15$$

From 6-7, the output current for that equation can be expressed as :-

$$i_{out} = gm_2 \frac{v_{id}}{2} + \left(-gm_4 v_m\right) \qquad 6\text{-}16$$

Replace v_m from 6-15 into 6-16, :-

$$i_{out} = gm_2 \frac{v_{id}}{2} + \frac{gm_1 \frac{v_{id}}{2}}{1+\frac{sC_n}{gm_3}} \qquad 6\text{-}17$$

Looking at the RHS of the small signal model, the current flow to output resistance parallel with output capacitance ($ro_4//ro_2//C_o$).

$$v_{out} = i_{out}\left(ro_4 \,//\, ro_2 \,//\, C_o\right) \qquad 6\text{-}18$$

Equation 6-18 can be written as :-

$$v_{out} = i_{out} \frac{1}{\frac{1}{ro} + sC_o} \qquad 6\text{-}19$$

Where *ro* is the output resistance, a parallel combination or ro_4 and ro_2.

Substitute 6-17 into 6-19, :-

$$v_{out} = gmro\frac{v_{id}}{2}\left(1+\frac{1}{1+\frac{sC_n}{gm_3}}\right)\left(\frac{1}{1+sC_o ro}\right) \qquad 6\text{-}20$$

The gain for the OTA can be expressed as :-

$$A_v = \frac{v_{out}}{v_{id}} = (gmro)\left(\frac{1}{1+sC_o ro}\right)\left(\frac{1+\frac{sC_n}{2gm_3}}{1+{sC_n}/{gm_3}}\right) \qquad 6\text{-}21$$

Looking at equation 6-21, the first part is the dc gain of the OTA which is represented by (gm $*$ ro). The second part represents the pole frequency, f_{p1} which is shown in equation 6-22. f_{p1} is the output pole which is dominant especially when a large load capacitance is present.

$$f_{p1} = \frac{1}{2\pi C_o ro} \qquad 6\text{-}22$$

The last part of equation 6-21 represent the second pole frequency and a zero frequency as shown in equation 6-23 and 6-24.

$$f_{p2} = \frac{gm_3}{2\pi C_n} \qquad 6\text{-}23$$

$$f_z = \frac{2gm_3}{2\pi Cn} \qquad 6\text{-}24$$

Once the small signal analysis has been done for both of the low frequency and high frequency effect, the behavioural model can be developed to include all the parameters from the small signal model. The behavioural description using Verilog-A behavioural language for the OTA is given in figure 6-11.

```
Module ota(inp, inm, out)
   ….
   parameter real gm = 60e-6;
   parameter real ro = 1e+6;
   electrical inp, inm, out, vm;
   real vin;
analog begin
   // high frequency model
   vin = V(inp,inm);
   I(vm) <+ -gm*(vin/2);  // gm transistor M1
   I(vm) <+ cin*ddt(V(vm)); // cin is the total input stage capacitance
   I(vm) <+ cgd1*ddt(vin/2);   // miller effect of cgd1
  …
   I(out) <+ -gm3*V(vm);
   I(out) <+ -gm*(vin/2);
   …..
   V(out) <+ I(out)*ro;
   …..
end
endmodule
```

Figure 6-11: Verilog-A code for OTA

To verify the accuracy of the behavioural model, a comparison is made between the behavioural model and transistor model for their frequency response. Figure 6-12 shows the response plot for behavioural model and transistor level simulation. As can be seen from the figure, the behavioural model matches the transistor level response with about 20% different. The different can be reduced by improving the behavioural model to include higher number of equations to model some other circuit parameters so that the response will match closely to the transistor level. However, this might affect the simulation time. Therefore, a trade off has to be made between accuracy and design speed.

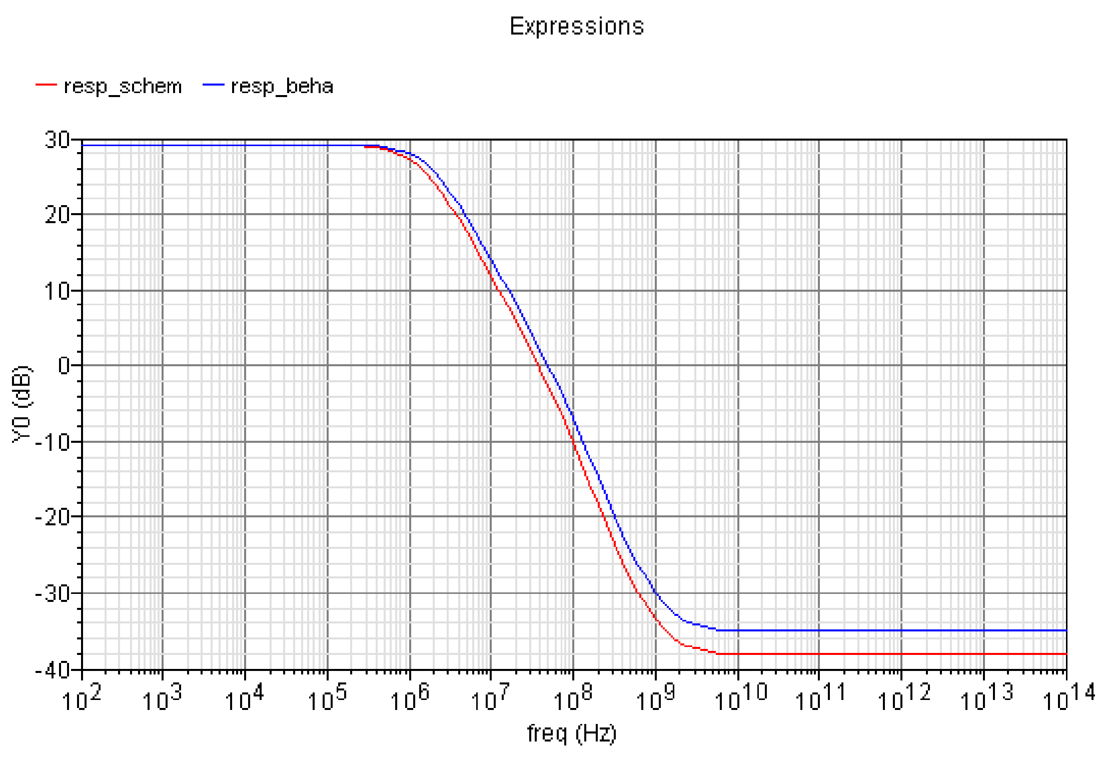


Figure 6-12: Comparison between behavioural model and transistor model

6.5.6 TDCD flow – System level optimisation

The behavioural OTA developed in previous section is instantiated in the filter system level description. The topology for the 7th order elliptic low pass filter is shown in figure 6-13. The designable parameters for the filter are OTA transconductance (gm) and all capacitor values (C1 ~ C10). The filter is optimised towards typical video filter specifications [94] as shown in figure 6-14 which defines the objective space of the optimisation.

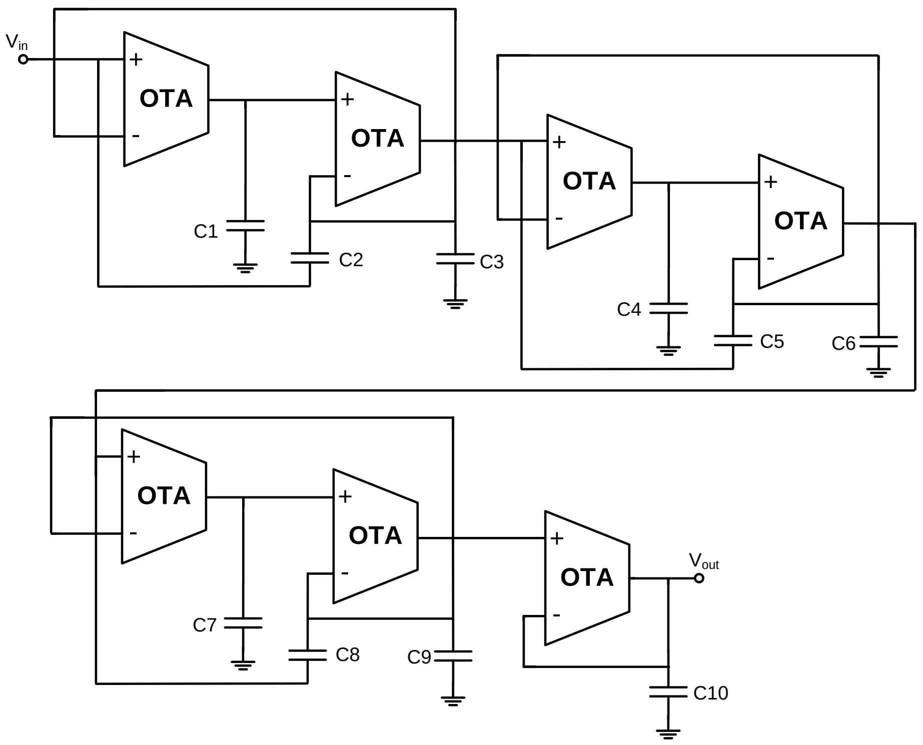


Figure 6-13: 7th order low pass elliptic filter

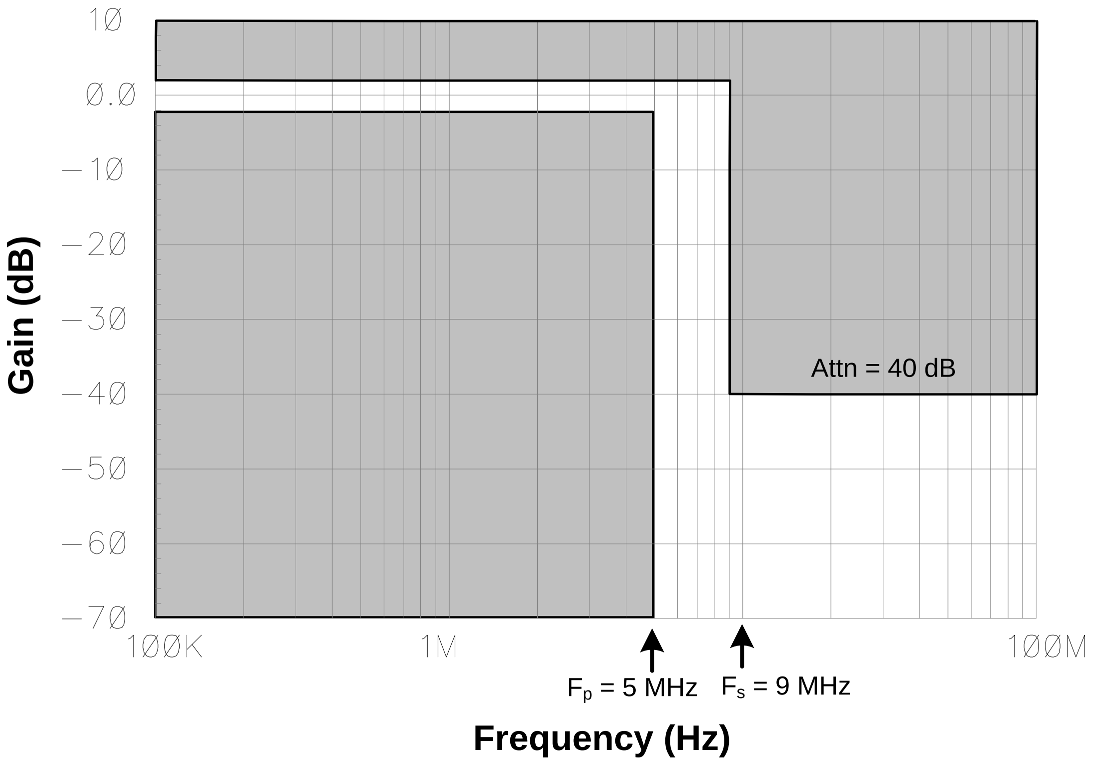


Figure 6-14: Filter specifications

A testbench was created to simulate the filter response. One testbench is sufficient to simulate all the performance functions required for the filter. Once the spice netlist has been created, a multi objective optimisation using NSGA-II algorithm is performed on the filter design to locate optimum solution points. A total of 200 individuals and 50 generations were used for the optimisation process. Some samples of the optimisation result are shown in table 6-2. This table shows all the design solutions that meet the filter specifications. In the next step, the design parameters of these solutions (i.e. gm) will be taken as the specification for the lower sub-block (OTA). This particular step in the design flow propagates the system level specifications to lower level sub-block. Once the lower level specification has been determined, the performance and variation model of the sub-block is used to search for the feasible and optimal solutions.

Based on the performance and variation model of the OTA (table 6-1), the only feasible solutions for the filter are design points 15 and 70 (refer to table 6-2). The other design points in the table require a higher transconductance value which is not feasible for the OTA topology.

Design:	**gm (μs) :**	**Attn (dB):**	**Fp(MHz):**	**Fs(MHz):**
11	122.3	40.3	6.1	8.3
22	131.6	47.4	5.4	7.5
15	108.9	45.9	5.3	7.3
70	113.8	55.1	5.7	8.9
61	130.4	61.7	5.7	8.9

Table 6-2: Pareto-front samples for filter optimisation

From table 6-2, looking at design point 15 and 70, the specifications for the OTA are 108.9u and 113.8u. The variation model of the OTA is used to interpolate the transconductance variation for these two values. For this example, the interpolated variation values for both of the transconductances are 0.75% and 0.73% respectively. These variations will be used to determine the minimum and maximum values for each of the transconductances. The minimum and maximum transconductance will be used in behavioural filter simulation to determine the filter performance with the effect of the variations. From the simulation, performances are compared with the specifications and the one that passes all the specifications will be chosen as the design solution. In this example, design point 15 and its variations pass all the filter specifications hence is chosen for the OTA design. The design parameters of the OTA will be interpolated from the transconductance value. The result of this hierarchical optimisation is a complete filter design that has been optimised to meet high level specifications taking process variations into consideration. To verify the predicted yield given by the proposed approach, a final Monte Carlo simulation with 100 samples was run on the transistor level filter design. This simulation confirmed a yield of 100% as shown in figure 6-15.

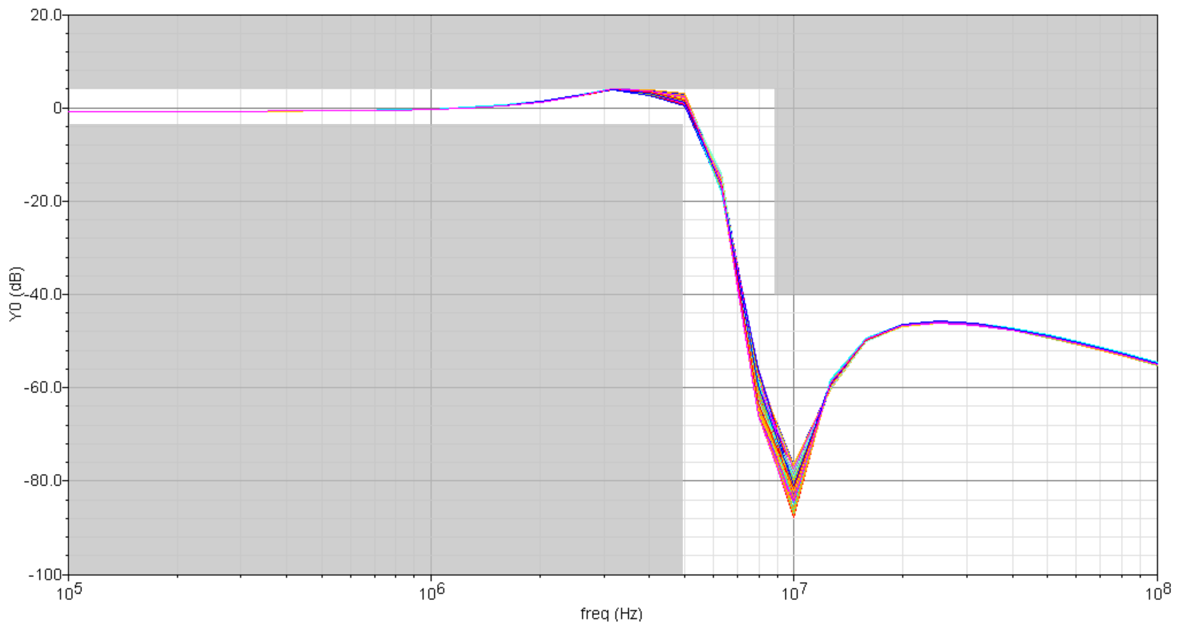


Figure 6-15: Monte Carlo plot of filter response

6.7 Summary

A new design flow for hierarchical-based circuit sizing is presented. The strategy combines a multi objective bottom up (MUBU) modelling to model individual sub-blocks and top down constrained design (TDCD) to break down the system level into sub-blocks and propagate the specifications. The new hierarchical-based design demonstrated how the performance and variation model developed in the MUBU stage can be exploited to predict the system level performance and its variations. This prediction is very useful to estimate and optimise the system yield. An example of 7th order low pass filter demonstrates the ability of the method to design and optimise the system for performances and yield.

Chapter 7

Mixed-signal System Level Application

7.1 Introduction

Chapter 6 has demonstrated the proposed methodology on a small system design. The example in that chapter shows the applicability of the method to find solution for small circuits with small design objectives. This chapter on the other hand will demonstrate the capability of the methodology to deisgn and optimise a bigger and complex mixed-signal system. A charge pump PLL that consists of a combination of analogue and digital block that requires higher number of SPICE analysis is used as the application example.

The PLLs is a typical analogue mixed signal system which plays an important role in many applications ranging from frequency generators to clock recovery in communication systems. Due to its mixed-signal nature, the design of PLLs becomes a crucial part of the time-to-market for many products. Simulating a PLL at transistor level takes a long time because of the large number of devices in the circuit. Also, the phase noise specification for the PLL requires transient noise simulation with a very well controlled time step and often takes considerable time to simulate. Due to this limitation, behavioural modelling was commonly used to model the individual blocks in PLL [95]. On top of that, hierarchical sizing methodology has been proposed to accelerate the design process of a PLLs [96, 97, 98].

A charge pump PLL consist of five building blocks: phase frequency detector (PFD), charge pump (CP), loop filter (LF), voltage controlled oscillator (VCO) and divider (D) as shown in figure 7-1. One of the application of PLLs is frequency synthesis. In a frequency synthesizer, the output frequency can be set to multiples of the reference input frequency (F_{ref}) by changing the divider ratio (N). The output frequency can be written as :-

$$F_{out} = N \times F_{ref} \qquad 7\text{-}1$$

The phase frequency detector (PFD) detects the phase and frequency difference between reference signal and the feedback signal from the divider. The charge pump (CP) transforms the phase difference of the PFD into output current. This current is delivered to the loop filter (LF) and the output of this filter is a control voltage (V_c) that control the VCO. The oscillation frequency of the VCO is determined by the control voltage. Once the feedback frequency match to the reference frequency, the control voltage become constant and the vco will oscillate at a constant frequency. This is the operation of PLL that is locked to a particular desired frequency.

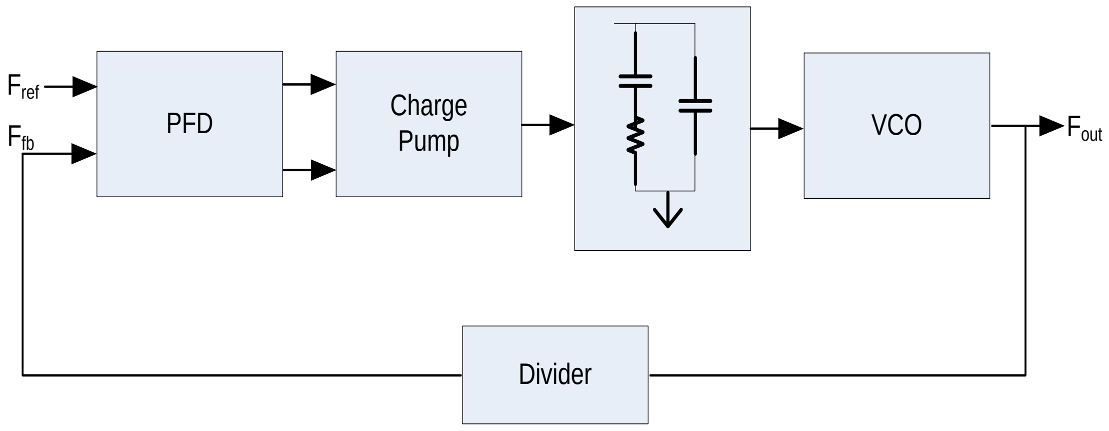


Figure 7-1: PLL system block diagram

In this chapter, the methods proposed in the previous chapter for performance and variation modelling and hierarchical-based opitmisation are used to efficiently design a complete PLL system. The process is divided into two stages: preparation stage for the performance and variation model development and design stage for the complete PLL system. The design and optimisation for the example will only consider the analogue blocks of the system namely the charge pump, loop filter and voltage-controlled oscillator (VCO), while the digital blocks are held as fixed.. The models that to be developed during the preparation stage are charge pump and VCO. The next section will briefly discuss the architecture of the PLL system.

7.2 PLL system

7.2.1 Phase Frequency Detector

The phase detector is a circuit whose the output is linearly proportional to the phase differece of its two inputs. Ideally, the relationship between output voltage (Vout) and phase difference (Δφ) is linear as depicted in figure 7-2. The slope of the line is the gain of the phase detector, K_{PD} and is expressed in V/rad.

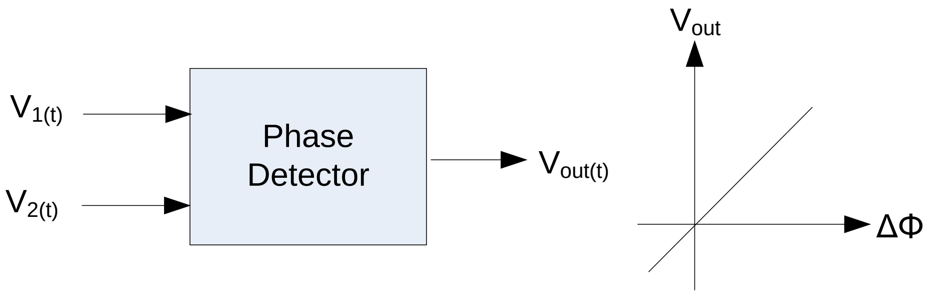


Figure 7-2: phase detector concept

A simple example of phase detector is the exclusive OR (XOR) gate as shown in figure 7-3. The plot shows how the width of the output pulses varies with the difference of the inputs.

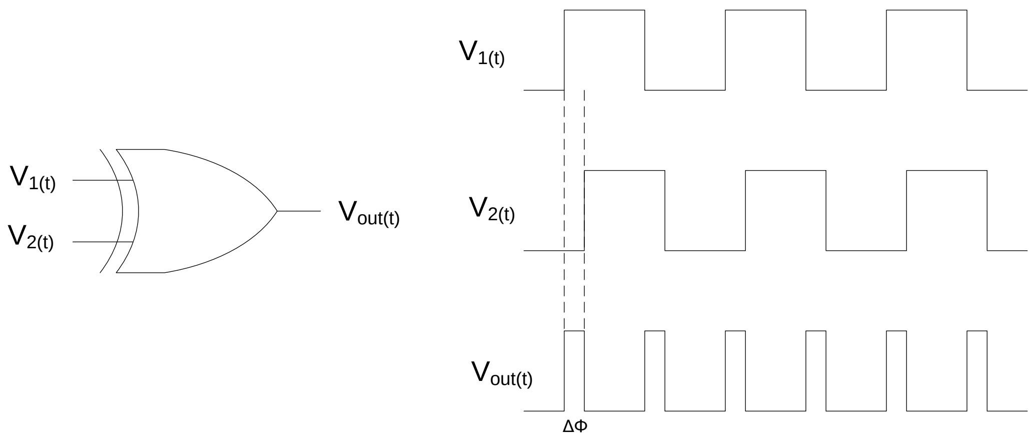


Figure 7-3: Phase detector plots

One of the main limitations of the phase detector is in its acquisition range [99]. The transition from the unlocked to the locked condition is nonlinear due to the inequality in the frequencies and the locking range is very limited. It is often necessary to have a wide acquisition range because the VCO oscillation frequency may vary considerably with process and temperature variation. Due to this limitation, a frequency comparison circuit is added to the phase detector so that the module can detect both the phase and frequency differences. This block is called phase/frequency detector (PFD) and a simple form of PFD circuit is illustrated in figure 7-4.

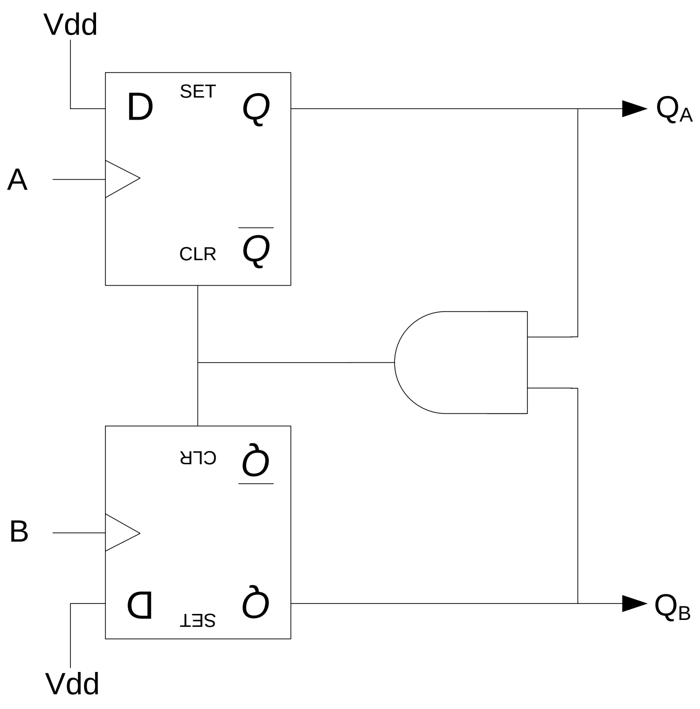


Figure 7-4: PFD schematic

7.2.2 Charge Pump and Loop Filter

A charge pump consists of two switched current sources: source and sink currents. Current charge is steered into or out of the loop filter in a PLL according to two logical inputs from PFD. Figure 7-5 illustrates a charge pump driven by PFD and driving a capacitor. If the PFD inputs (Q_A and Q_B) are the same (no difference in phase and frequency of signals A and B), switch S_1 and S_2 are off and V_{out} remains constant. If Q_A is high and Q_B is low, then I_1 will be steered to capacitor C_p (current is steered into the loop filter) and if Q_B is high and Q_A is low, I_2 will discharge the capacitor (current is steered out of the loop filter). The plot in figure 7-5 shows the rising up of V_{out} when signal A leads signal B.

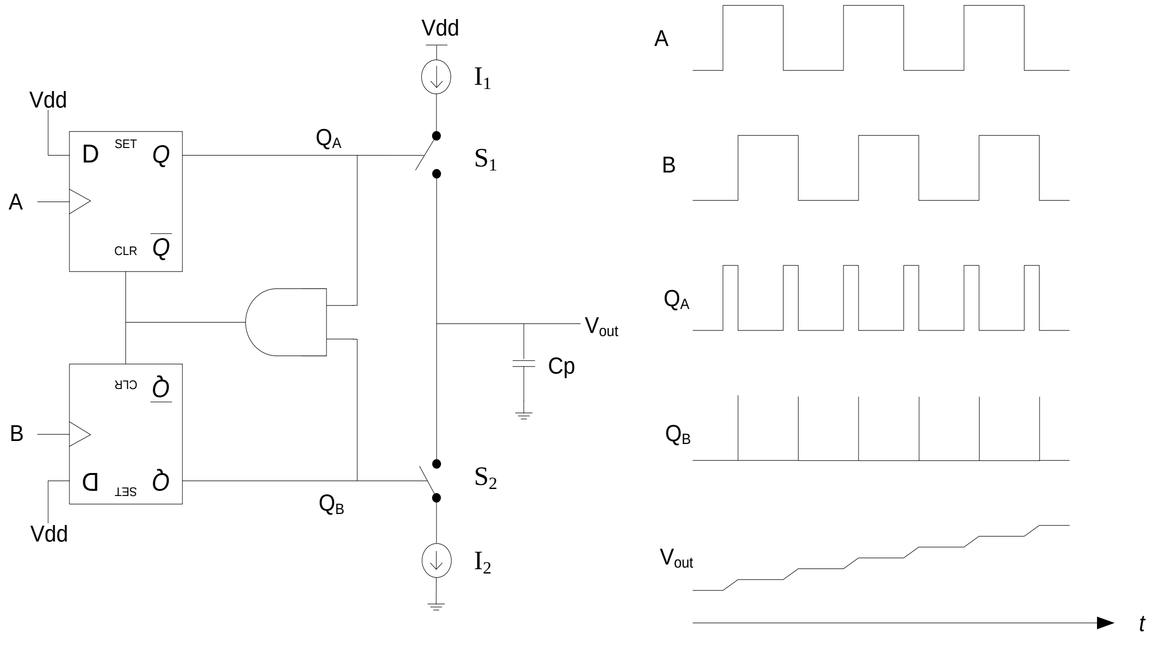


Figure 7-5: PFD/CP illustration and its signal plots

In figure 7-5, a capacitor, C_p is used in place of the filter. The loop filter for a PLL can be made from a simple RC filter. Figure 7-6 shows a 2nd order RC filter that is commonly used in a PLL system. The filter is composed of a resistor R1 in series with a capacitor C1. the charge pump current sources and the capacitor form an integrator and the resistor introduces a zero point of the system. However, this configuration will introduce a ripple of $I_{pump}R_1$ on the output voltage, V_{out} and this ripple modulates the

VCO output frequency and may cause excessive jitter. In order to suppress this ripple or voltage spike, a small capacitor C_2 is added in parallel with R_1 and C_1. In practical design, C_2 is usually chosen to be about $C_1/10$. A small C_2 improves the phase margin of the PLL system.

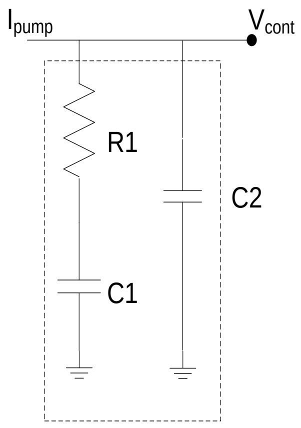


Figure 7-6: Loop filter

7.2.3 Voltage Controlled Oscillators

Oscillators play an important role in phase locked loop system. In general, a simple oscillator produces a periodic output, usually in the form of voltage. In PLLs, the oscillator is required to be tuneable i.e., the frequency oscillation is a function of a control input, usually a voltage hence the name voltage-controlled oscillators (VCO). An ideal voltage-controlled oscillator is a circuit that generates a periodic signal whose the frequency is a linear function of its control voltage, as illustrated in figure 7-7. This linear relationship is expressed in equation 7-2.

$$f_{out} = f_{\min} + K_{VCO}.(V_{in} - V_{\min}) \qquad 7\text{-}2$$

Where f_{out} is the output frequency, f_{min} is the minimal frequency, V_{in} is the output voltage from loop filter and V_{min} is the minimum input voltage. K_{VCO} denotes the gain of the circuit which can be defined as in equation 7-3.

$$K_{VCO} = \frac{(f_{\max} - f_{\min})}{(V_{\max} - V_{\min})} \tag{7-3}$$

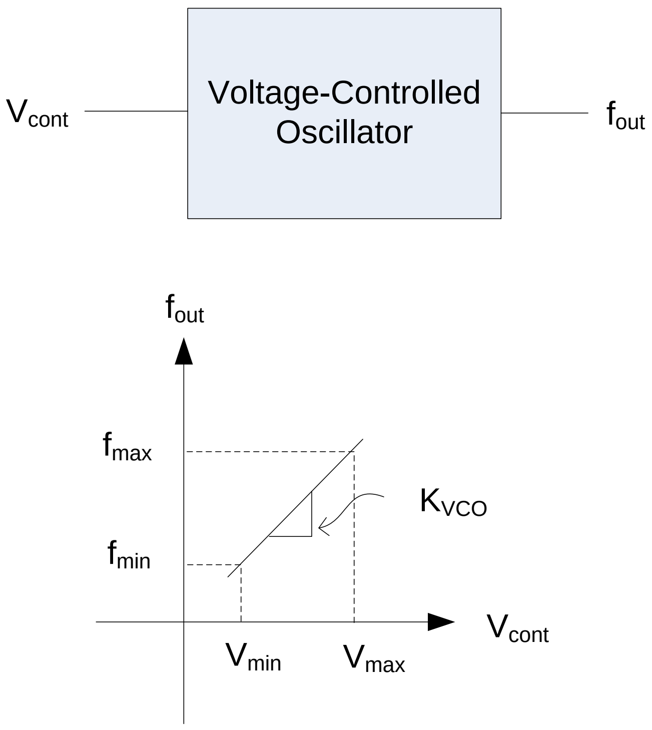


Figure 7-7: VCO as a linear function of control voltage

7.3 PLL System Performances

A PLL system is usually designed to meet several requirements for a particular application. For example, a frequency synthesizer may require a PLL to have a better locking time, low phase noise, low power consumption, better stability and operate at a wide tuning frequency range. Some of the performances commonly associated with PLLs will be discussed in the remainder of this section. The discussion will be divided into two sections : first section will discuss a group of PLL performances that can be represented by the PLL transfer function such as loop bandwidth, locking time and phase margin and the second sectin will discuss about PLL phase noise parameters, extracting individual noise and behavioural noise modelling.

7.3.1 PLL transfer function

When the PLL is said to be in-lock, it can be represented in s-domain block diagram as shown in figure 7-8. K_{PD} is the gain of phase/frequency detector which is given by equation 7-4, K_{VCO} is the gain of the VCO and F(s) is the transfer function of the loop filter. The open-loop transfer function of this model is represented by equation 7-5.

$$K_{PD} = \frac{I_{CP}}{2\pi} \qquad \text{7-4}$$

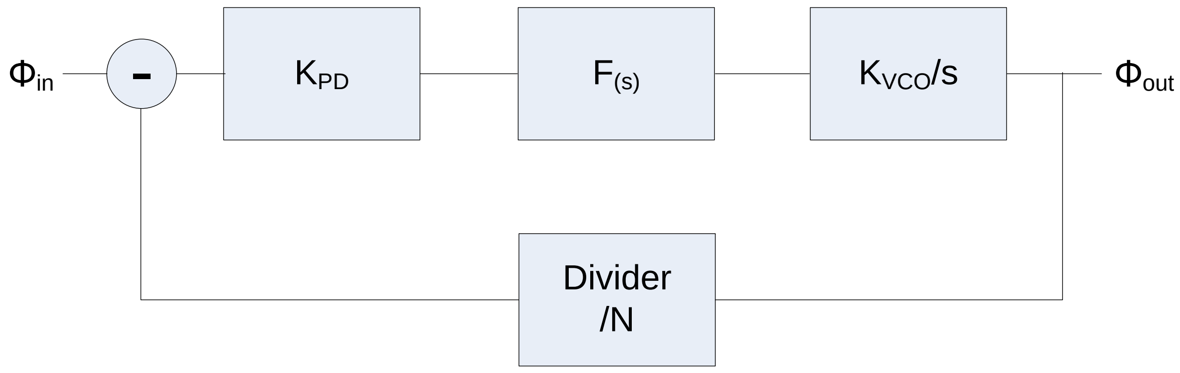


Figure 7-8: A linear PLL model

$$\frac{\phi_{out}}{\phi_{in}}\Big| open = K_{PD}F_{(S)}\frac{K_{VCO}}{s} = \frac{I_{CP}}{2\pi}\frac{RC_1 s + 1}{RC_1C_2s^2 + (C_1 + C_2)s}\frac{K_{VCO}}{Ns} \qquad \text{7-5}$$

The system has a zero at :

$$w_z = \frac{1}{RC} \qquad \text{7-6}$$

Based on equation 7-5, the close loop transfer function can be written as :

$$\frac{\phi_{out}}{\phi_{in}}\Big|_{close} = \frac{\frac{K_{VCO}I_{CP}R}{2\pi N}\left(s + \frac{1}{RC_1}\right)}{s^2 + \frac{K_{VCO}I_{CP}R}{2\pi N}s + \frac{K_{VCO}I_{CP}}{2\pi NC_1}} = \frac{2\zeta\omega_n s + \omega_n{}^2}{s^2 + 2\zeta\omega_n s + \omega_n{}^2} \qquad \text{7-7}$$

Where ω_n is the natural frequency and ζ is the damping factor. From this equation, the natural frequency and damping factor can be expressed by equation 7-8 and 7-9.

$$\omega_n = \sqrt{\frac{I_{CP}K_{VCO}}{2\pi NC_1}} \qquad 7\text{-}8$$

$$\zeta = \frac{\omega_n RC_1}{2} \qquad 7\text{-}9$$

The loop bandwidth can be expressed by equation 7-10.

$$\omega_{BW} = \omega_n \sqrt{\left(1 + 2\zeta^2 + \sqrt{2 + 4\zeta^2 + 4\zeta^4}\right)} \qquad 7\text{-}10$$

From equation 7-10, it can be seen that the loop bandwidth is determined by K_{VCO} from the VCO block, I_{CP} from the charge pump block, C_1 and R from the loop filter block. The locking time for a PLL system, according to [100] is given by equation 7-11.

$$T_{Lock} = \frac{2\pi}{\omega_n} \qquad 7\text{-}11$$

The bode plot for the open loop transfer function given in equation 7-5 is illustrated in figure 7-9. The unity gain bandwidth is the value of the frequency when the magnitude of the open loop gain is 1 and can be expressed by equation 7-12. The bode plot has a pole given by equation 7-13.

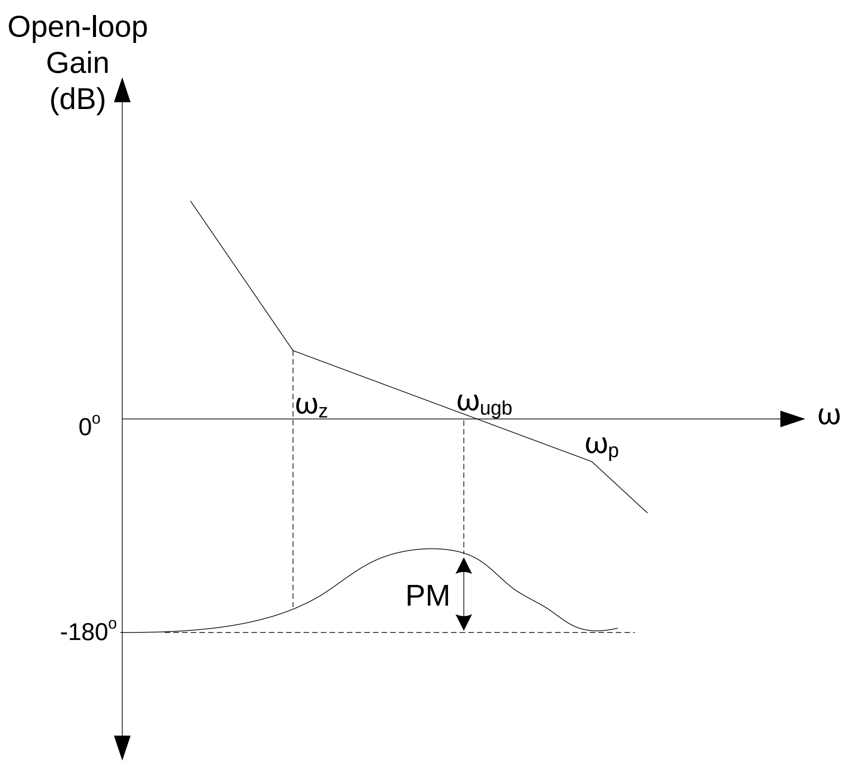


Figure 7-9: Bode plot of a 3rd order PLL

$$\omega_{UGB} = \frac{I_{CP}}{2\pi}.\frac{K_{VCO}}{N}.R \qquad 7\text{-}12$$

$$\omega_p = \frac{1}{R(C_1 // C_2)} \qquad 7\text{-}13$$

The phase margin (PM) of the system which is used to determine the stability can be calculated using equation 7-14.

$$PM = \arctan\left(\omega_{UGB} / \omega_Z\right) - \arctan\left(\omega_{UGB} / \omega_p\right) \qquad 7\text{-}14$$

Until this point, this section has discussed several PLL performances such as loop bandwidth, damping factor, natural frequency, phase margin and locking time. In the top PLL system, these performances are evaluated analytically using all the equations discussed earlier in order to determine the PLL performances. Another important performance function of the PLL system is phase noise or jitter (in time domain) and this will be discussed next.

7.3.2 PLL phase noise

Every building block of the PLL will contribute to the total output noise, which is characterized in terms of phase noise in the phase domain or jitter in the time domain [101]. Figure 7-10 shows a PLL system with all the noise contributions from each block. The noise sources N_{ref}, $N_{PFD/CP}$, N_{LF}, N_{VCO} and N_{Div} are placed respectively at the corresponding nodes.

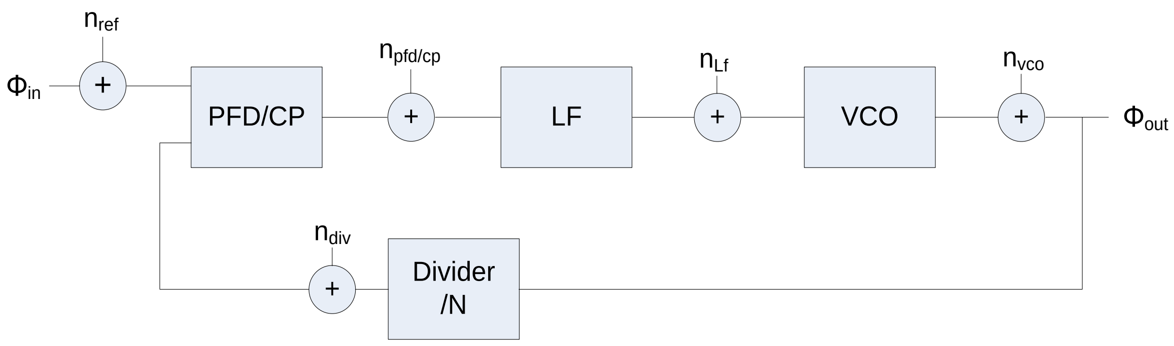


Figure 7-10: Noise analysis model for PLL system

In the PLL system, each block can be considered to have an individual effect to the output noise and from all the individual noise sources, a superposition can be applied to compute the total PLL output noise [102]. Each noise source can be derived as a laplace transfer function that represents how the PLL output noise is shaped by them. The noise transfer function originating from the reference oscillator, divider and PFD/CP block will have a low pass response. Therefore the PLL output phase noise will be strongly effected by the phase noise of these blocks at low offset frequencies. The noise transfer function between output and VCO input tends to be a high pass response and therefore the phase noise of the PLL output due to the VCO phase noise will be affected at the high offset frequencies. For the loop filter, the injected noise has a band pass response and will shape the PLL noise accordingly. The closed-loop phase noise of the PLL ($L_{PLL}(f)$) can be computed by performing a superposition over each of the contributing noise sources with the assumption that no correlation exists between them.

7.3.3 Extracting individual phase noise contribution

In order to calculate the PLL output noise, the phase noise contribution from each of the individual block must be analysed and extracted. This can be done through a spice phase noise analysis for each of the block separately. For the scope of this thesis, only noise contribution from VCO block and PFD/CP+filter block will be considered and this is discussed next.

7.3.3.1 VCO Noise

In most applications, the PLL phase noise is dominated by VCO phase noise [103]. This is because oscillators tend to amplify noise found near their oscillation frequency and any of its harmonics. To extract the phase noise parameter of a VCO, a phase noise analysis is done using a RF simulator such as SpectreRF [10] or HspiceRF [9]. The phase noise, *L* is measured for a range of frequencies offset from the centre frequency [104, 105]. A graph for the phase noise value versus the offset frequencies is as illustrated in figure 7-11. If flicker noise is present, there will be a range of low frequencies for which the power noise drops at a rate of 30dB per decade. Above this, the rate of drop will be 20dB per decade which is characterized as white noise region. All these information from this plot (flicker and white noise) is extracted and will be used in behavioural description of VCO to represent the phase noise slope.

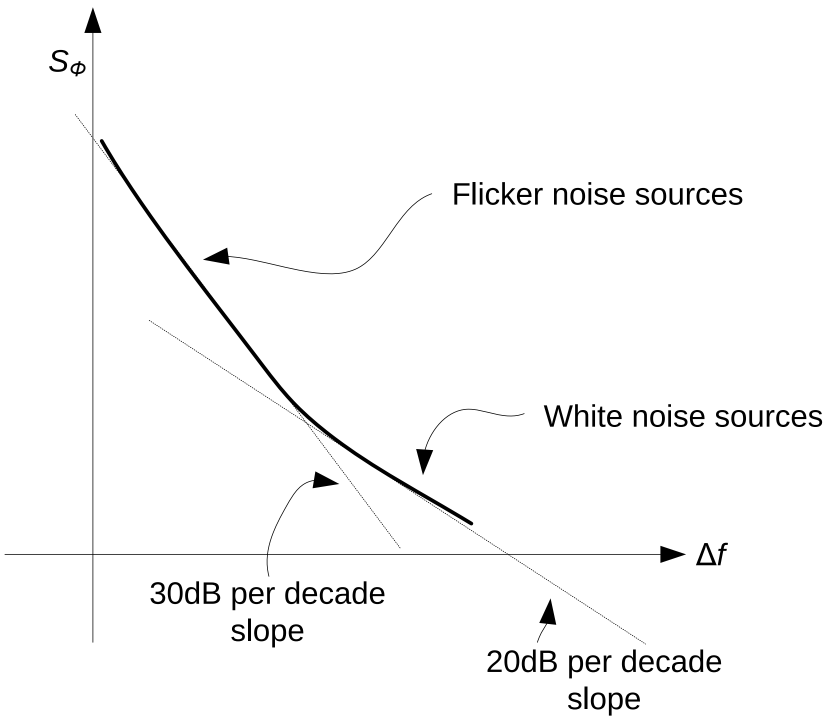


Figure 7-11: VCO phase noise vs offset frequencies

7.3.3.2 PFD/CP and loop filter noise

Other than VCO noise, the noise combination of the phase frequency detector, the charge pump and the loop filter also contribute to the PLL noise. The combination noise can be extracted by simulating the PFD, charge pump and loop filter under open loop conditions that approximate the PLL in a locked steady-state. The schematic for this analysis is shown in figure 7-12. In this schematic, the phase frequency detector is driven with an in-phase clock to represent a locked-state of PLL.

The output noise from the simulation can be extracted and represents another noise contribution in the PLL closed-loop system analysis. Figure 7-13 shows an illustration of the noise simulation result for PFD/CP and loop filter combination.

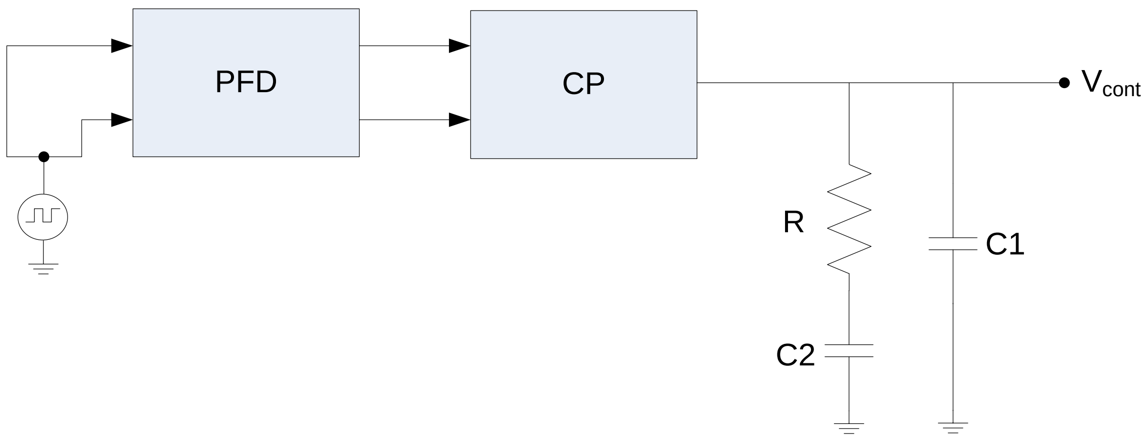


Figure 7-12: A schematic for PFD/CP and loop filter noise simulation

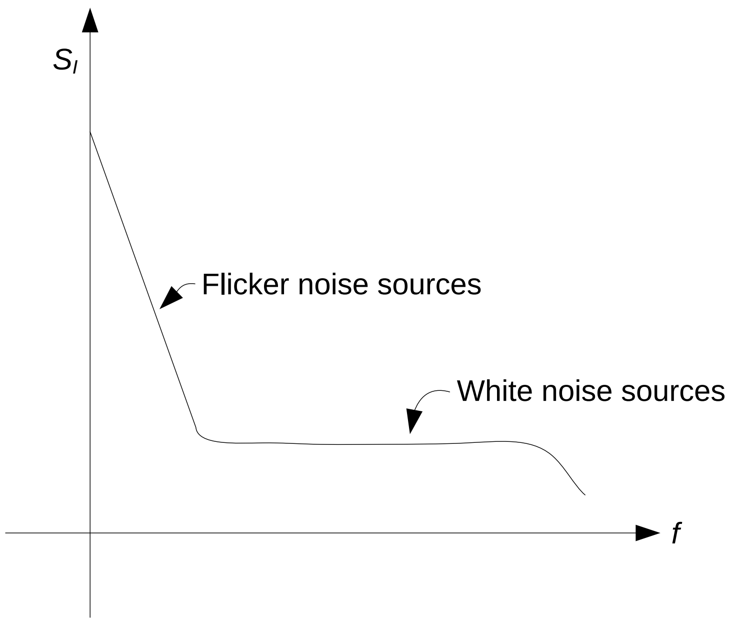


Figure 7-13: Illustration of PFD/CP and loop filter noise plot

7.3.4 Behavioural Modelling of Noise Sources

Once all the noise contributions from individual blocks have been calculated, their values can be represented using a behavioural model. Verilog-A provides a flicker_noise function for modelling transitor model flicker noise, which has a power spectral density proportional to $1/f^{\alpha}$ with α typically close to 1. However, Verilog-A does not limit the value of α, making the function well suited to model the oscillator phase noise with α=2 for 20dB roll off noise and α close to 3 for 30dB roll off noise. [106].

Typically the VCO phase noise contributions $L_{VCO}(f)$ can be modelled with a frequency dependent phase noise expression given in equation 7-15.

$$L_{VCO}(f) = \frac{K_W}{f^2} + \frac{K_F}{f^{2+EF}} \qquad \text{7-15}$$

Where EF ≈ 1 is the flicker noise exponential used within the transistor models, K_F represent modulated flicker noise contributions and K_W represent modulated white noise contributions. The PFD/CP $L_{PFD/CP}(f)$ noise response can be modelled with a frequency dependent phase noise expression given in equation 7-16.

$$L_{PFD/CP}(f) = \frac{K_F}{f^{EF}} + K_W \qquad \text{7-16}$$

Where K_F represent flicker noise contribution and K_W represent white noise contribution.

With all the noise contribution transfer functions obtained above, a behavioural model can be developed based on the expressions given in equation 7-15 and 7-16. One of the advantages of Verilog-A is its ability to model both signal and noise characteristics within the same module. In this way, the noise is modelled by adding noise voltages to the voltage variables. Figure 7-14 shows a Verilog-A module for modelling the phase domain VCO with noise. The phase signal model of a VCO is an ideal integrator that converts frequency to phase based on the VCO gain providing a transfer function as given in equation 7-17. A Verilog-A laplace transform operator, *laplace_nd* is used to represent the transfer function. Added to the output voltage of the VCO are two flicker noise function, *flicker_noise( ),* that add f^3 to represent the 30dB roll off and f^2 to represent 20dB roll off noise distibutions.

$$H_{VCO}(s) = \frac{K_{VCO}}{s} \qquad \text{7-17}$$

Figure 7-15 shows the Verilog-A code for implementing the PFD/CP behavioural model. The output of the block is modelled by a simple constant gain coefficient, K_d, that operates on the input different , V_{in1}-V_{in2}. As with the VCO model, the noise voltages of flicker noise and white noise are added to the output of the PFD/CP module.

```
// VCO behavioural model incorporating noise transfer function
Module vco()
...
V(out) <+ laplace_nd(V (in), {fmax-fmin/1},{0,1})
  + flicker_noise(lffl, 3, "VCO_flicker")
  + flicker_noise(lfwh, 2, "VCO_white");

end
endmodule
```

Figure 7-14: VCO behavioural model

```
// PFD/CP behavioural model incorporating noise transfer function
Module pfdcp
…
V(out) <+ kd*(V(in1) - V(in2))
  + flicker_noise(lfpfl, 1, "pfd_flicker")
  + white_noise(lfpwh, "pfd_white");

end
endmodule
```

Figure 7-15: PFD/CP behavioural model

In figure 7-14 and 7-15, Lffl, Lfwh, Lfpfl and Lfpwh are the VCO flicker noise, VCO white noise, PFD/CP flicker noise and PFD/CP white noise contribution respectively. With all the models for individual blocks developed, a top-level closed-loop PLL noise analysis can be performed. The PLL phase noise plot from the top-level simulation is shaped by the combination of the individual noise sources.

7.4 Design Example

A charge pump PLL system was designed using ST 0.12µm process technology and a supply voltage of 1.2V. The Multi Objective Bottom Up (MUBU) + Top Down Constraint Driven (TDCD) hierarchical design methodology discussed in chapter 6 was used to design the complete PLL system. The specifications for the PLL are given in table 7-1 and the system level block diagram is as illustrated in figure 7-1. The PLL was designed to generate frequency range of 500MHz to 1.2GHz from a 50 MHz reference oscillator. Therefore the divider ratio can be selected between 10 to 24 for the output frequency range. Only the analogue blocks of the charge pump (CP), VCO and the loop filter (LF) are considered in the design process while the digital blocks (PFD and Divider) are assumed to be ideal and held as fixed. As explained in chapter 6, the design methodology starts with multi-objective bottom up modelling to model the performance and variation of the sub-blocks using the methodology proposed in chapter 5. The work can be divided into 2 stages : preparation stage for the model development and design stage for the whole PLL system. In the preparation stage, 2 sub-block models were developed using the MUBU technique. Both of these models were used later for the PLL design using the TDCD method.

Performances	**Specifications**
Output Frequency Range	500MHz to 1.2GHz
Locking time	< 1us
Current consumption	< 5mA
Phase noise (@ 1 MHz offset)	< -100 dBc/Hz
Phase Margin	> 45 degress

Table 7-1: PLL system level specifications

7.4.1 Charge Pump (CP) performance and variation model

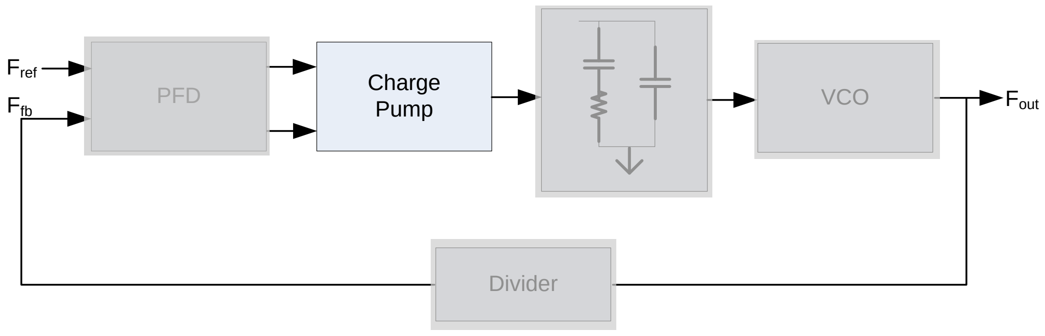


Figure 7-16: Charge pump preparation stage

The charge pump is a circuit that is used to steer the current into or out of loop filter based on the up and down signal from phase frequency detector. The schematic for an externally-biased charge pump is given in figure 7-17. The $up, down$, $\overline{up}$ and $\overline{down}$ are coming from PFD circuit. When the up signal is active, the current flows into the loop filter and causes the output voltage to rise up which in turn forces a higher oscillation frequency from VCO. On the other hand, when the down signal is active, the current flow out of the loop filter and causes the output voltage drops down and force a lower oscillation frequency. A dummy switch is added in this design in order to reduce the charge spike during switching.

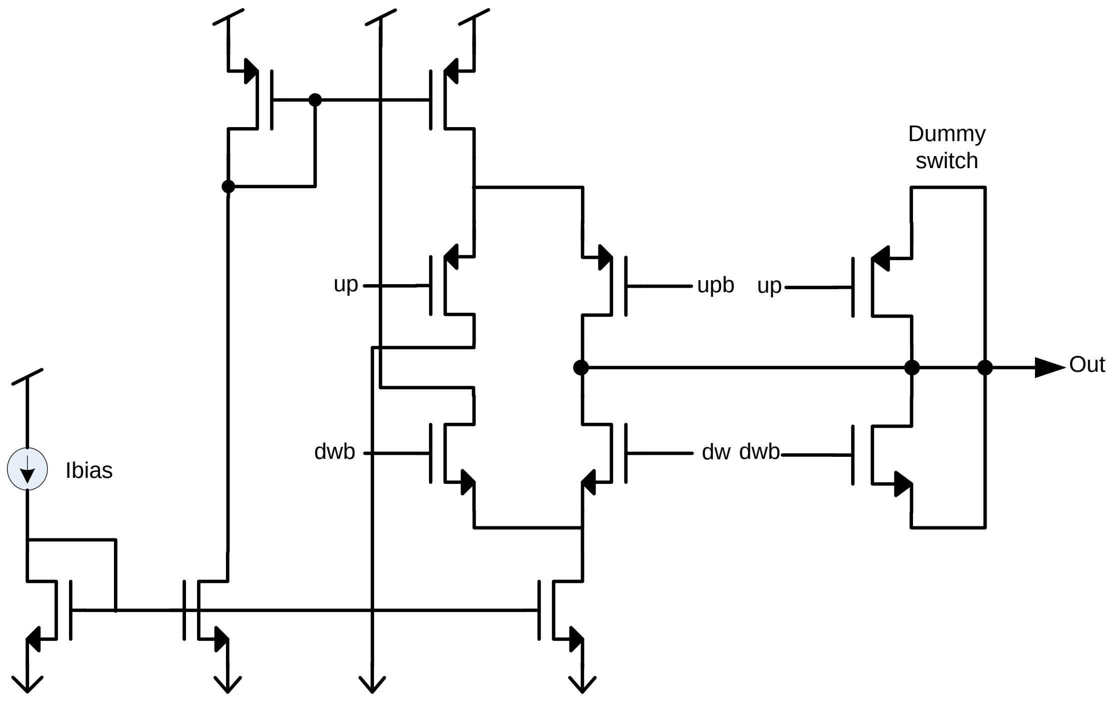


Figure 7-17: Charge pump (CP) schematic diagram

For the charge pump, two performance functions are being evaluated: charge pump current and output noise voltage. Multi objective optimisation was performed to the design using NSGA-II algorithm [51] in order to search for optimum performance trade-offs. A total of 30 generations with a population size of 50 were used giving a total of 1,500 samples. The outcome of this optimisation is a set of Pareto-points that represent the trade-off between competing performance objectives. All the points on this Pareto-front are stored in a lookup table which represent the performance model for the charge pump.

The next step is to develop a variation model based on the performance Pareto-points. All the points on the Pareto front undergo a Monte Carlo simulation using ST 0.12μm process variation and mismatch model. A 30 samples Monte Carlo simulation was performed on each of the Pareto points. The outcome of this Monte Carlo simulation is a set of performance variations deviated from its nominal value. For example, figure 7-18 shows the nominal, minimum and maximum plots for charge pump noise voltage for one of the Monte Carlo simulations. In order to estimate the minimum and maximum region for the performance functions, the standard deviation of the samples is calculated and this value is multiplied by 6 in order to get the 6 standard deviation range from the mean value. Figure 7-19 shows the Pareto plot of the performances (from MOO) and their variation obtained from Monte Carlo simulation. From the plot, the Pareto front clearly shows the trade off between current consumption and output noise voltage which indicates that a larger current will result in a smaller noise voltage. In addition to that, the minimum and maximum Pareto show how the nominal points will deviate due to the process variation and circuit mismatch.

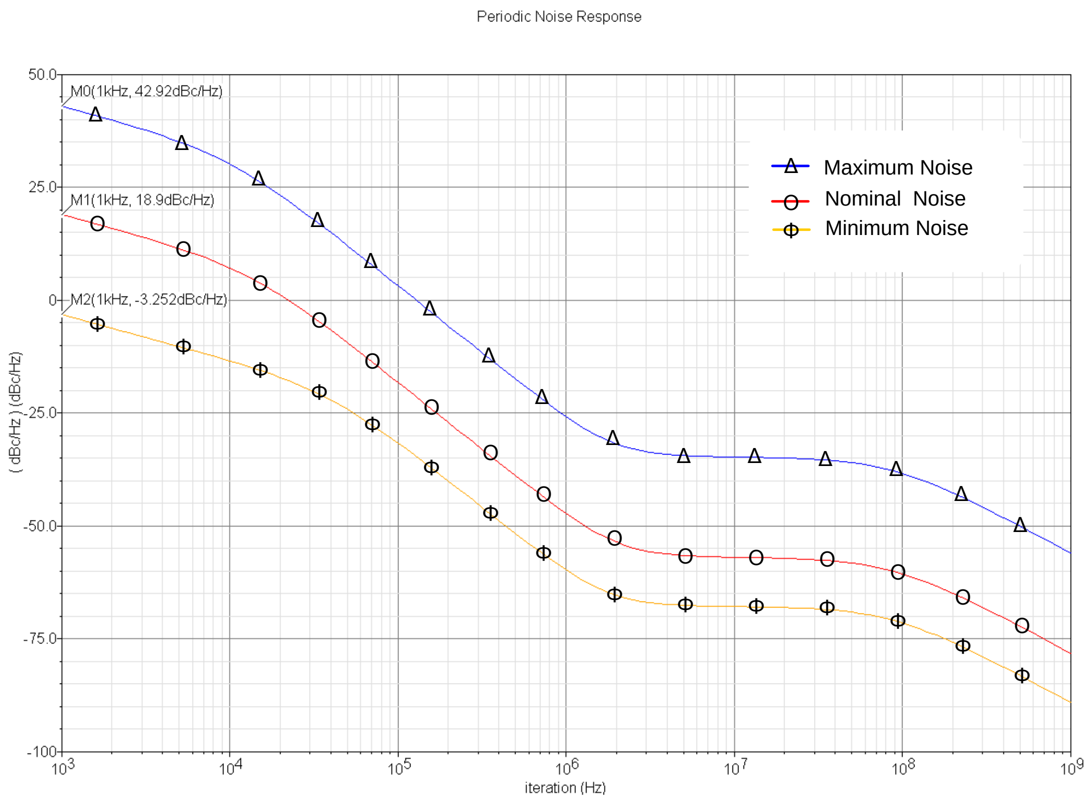


Figure 7-18: Nominal, minimum and maximum plot for charge pump noise

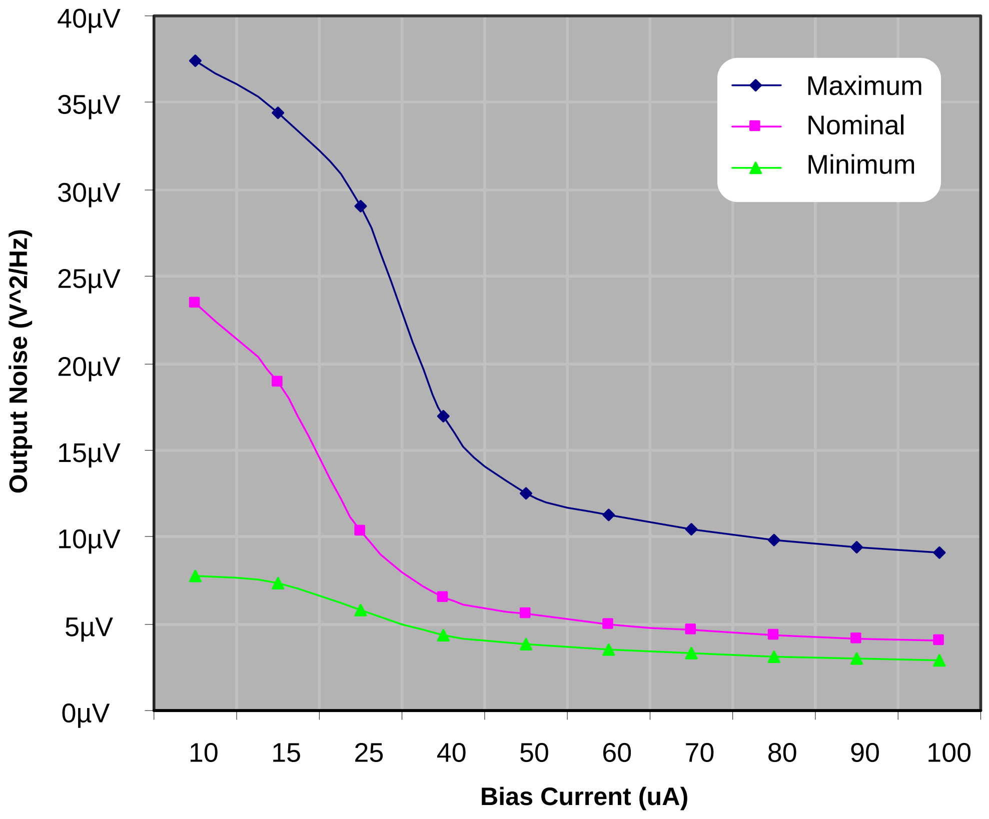


Figure 7-19: Charge Pump Pareto Front with Variations

The minimum and maximum performances obtained from MC analysis are stored in another look up table. At this stage a complete performance and variation lookup table has been developed for the PFD/CP and can be used in Verilog-A table model function for the behavioural modelling of the charge pump circuit. A part of the behavioural model incorporating the table model function nominal, minimum and maximum performances are shown in figure 7-20,7-21 and 7-22 respectively.

In the nominal performance behavioural model (figure 7-20), the table model function of the Pareto-front is used to interpolate the output noise, *lfpwh*, from a chosen bias current, *Icp*. The minimum and maximum behavioural models (figure 7-21 and 7-22) are used to interpolate and determine the performance variations (lfpwhmin and lfpwhmax) of the nominal performances.

```
Module pfdcp_nom
analog begin
…
//lookup table for pfd_cp noise
lfpwh = $table_model(Icp, "pfd_data.tbl","3E");

V(out) <+ kd*(V(in1) - V(in2))
  + flicker_noise(lfpfl, 1, "pfd_flicker")
  + white_noise(lfpwh, "pfd_white");

end
endmodule
```

Figure 7-20: PFD/CP table model function for nominal performances

```
Module pfdcp_min
…
…
//lookup table for pfd_cp noise
lfpwh = $table_model(Icp, "pfd_data.tbl","3E");

//lookup table for pfd variation
lfpwhmin = $table_model(lfpwh, "pfdmin_data.tbl", "3E");

V(out) <+ kd*(V(in1) - V(in2))
  + flicker_noise(lfpfl, 1, "pfd_flicker")
  + white_noise(lfpwhmin, "pfd_white");

$fclose(file_ptr1);

end
endmodule
```

Figure 7-21: PFD/CP table model function for minimum performances

```
analog begin
…
//lookup table for pfd_cp noise
lfpwh = $table_model(Icp, "pfd_data.tbl","1E");

//lookup table for pfd variation for maximum
lfpwhmax = $table_model(lfpwh, "pfdmax_data.tbl", "3L");

V(out) <+ kd*(V(in1) - V(in2))
  + flicker_noise(lfpfl, 1, "pfd_flicker")
  + white_noise(lfpwhmax, "pfd_white");

$fclose(file_ptr1);

end
endmodule
```

Figure 7-22: PFD/CP table model function for maximum performances

7.4.2 Voltage-controlled Oscillator (VCO) performance and variation model

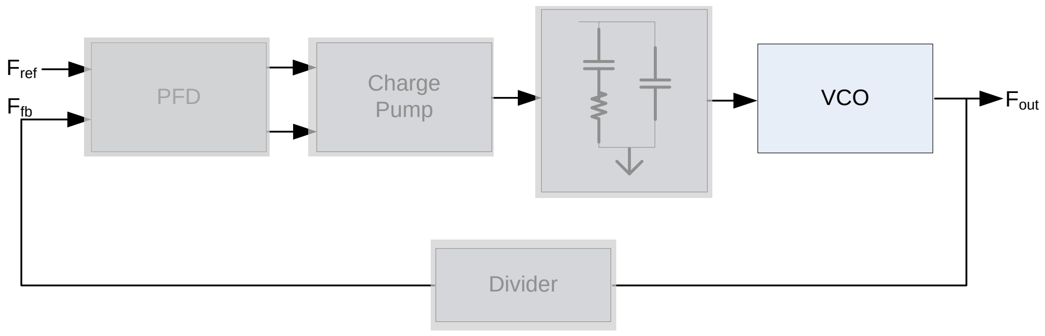


Figure 7-23: VCO preparation stage

The VCO is one of the important blocks in PLL system and a major contributor to PLL phase noise [81]. The chosen VCO topology is a 5 stage ring oscillator as shown in figure 7-24. In this kind of VCO, the input voltage controls the current through the delay cells which determines the delay time of each stage hence controlling the output oscillation frequency. An ideal VCO generates a periodic signal whose frequency is a linear function of the controlling voltage as explained earlier in this chapter.

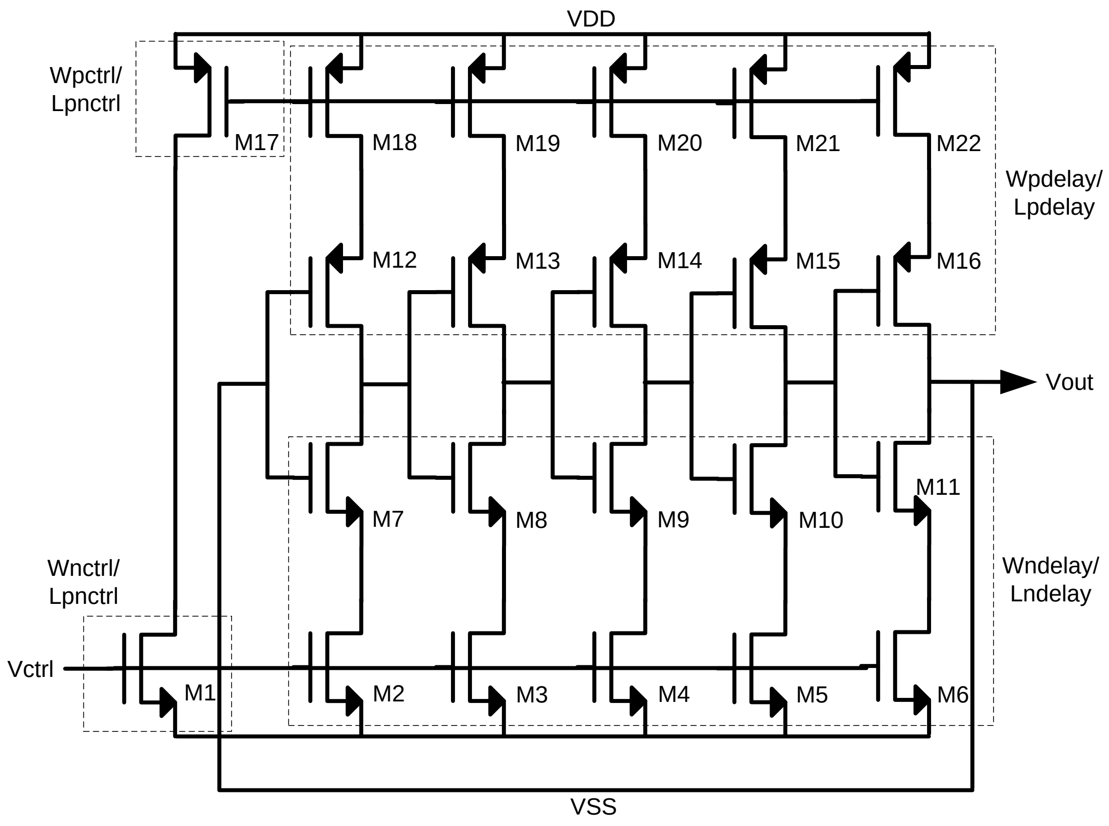


Figure 7-24: 5-stage ring VCO schematic

The first step in multi objective optimisation for the VCO is to determine the designable parameters for the circuit. In this example, these include the transistor lengths and widths making a total of 7 designable parameters. The parameters are shown in table 7-2 and illustrated by dotted line in figure 7-24. The performance functions for which the Pareto front must be generated are VCO phase noise, current consumption, VCO gain, minimum frequency and maximum frequency. A testbench netlist was created to evaluate these performance functions.

Block	Design Parameters	Range
Control	Length of M17 & M1	0.12µm – 1µm
	Width of M17	10µm – 100µm
	Width of M1	
Delay Cell	Width of all PMOS	
	Width of all NMOS	
	Length of all PMOS	0.12µm – 1µm
	Length of all NMOS	

Table 7-2 Design Parameters

The designable parameters must be constrained within a reasonable range (based on the targeted active area of the circuit) which defines the design space of the optimisation. In this example, all transistor lengths and widths were specified to be between 0.12µm-1µm and 10µm-100µm respectively as can be seen in table 7-2. A GA string is constructed based on the designable parameters as shown in figure 7-25 and will be used by the NSGA-II algorithm to generate the parameters for the spice simulation. A total of 30 generations each with a population size of 100 were used in this example, giving a total of 3,000 samples for the optimisation.

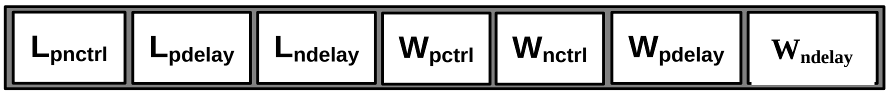


Figure 7-25: VCO GA string

The testbench netlist is used to evaluate each of the performance functions for every design parameter set generated by GA and the result of the simulations determines the fitness score of the individual sets. A non-dominated sorting and crowding distance method of NSGA-II (as explained in chapter 3) was applied to the solutions to determine the final set of Pareto-fronts.

From the MOO, a set of optimal solutions known as Pareto-fronts for the VCO was obtained. Table 7-3 shows some samples from the Pareto-points and table 7-4 shows the design parameters for those samples. All the points on the Pareto-front are the best trade-off for the design for all of the competing objectives. All the points on the Pareto-fronts and their corresponding design parameters represent the performance model for the VCO and are stored in a data file.

Design:	**Performance functions**				
	Min. Freq	**Max. Freq**	**VCO gain**	**VCO jitter**	**VCO current**
1	80.3 MHz	568 MHz	487 MHz	9.46 ps	3.09 mA
2	130 MHz	760 MHz	630 MHz	8.66 ps	3.10 mA
3	147 MHz	906 MHz	758 MHz	7.33 ps	3.71 mA
4	183 MHz	843 MHz	659 MHz	0.83 ps	2.12 mA
5	204 MHz	657 MHz	453 MHz	0.36 ps	1.79 mA
6	217 MHz	2.04 GHz	1.83 GHz	0.71 ps	4.99 mA
7	222 MHz	808 MHz	586 MHz	0.37 ps	3.41 mA
8	238 MHz	1.41 GHz	1.17 GHz	0.33 ps	7.64 mA
9	284 MHz	1.20 GHz	917 MHz	0.43 ps	4.67 mA
10	312 MHz	2.67 GHz	2.36 GHz	0.34 ps	6.53 mA

Table 7-3: Pareto-point samples for VCO

Design:	**Design Parameters**						
	L_{pnctrl}	L_{pdelay}	L_{ndelay}	W_{pctrl}	W_{pdelay}	W_{nctrl}	W_{ndelay}
1	0.56 µm	0.38 µm	0.16 µm	88.78 µm	12.28 µm	10.09 µm	70.95 µm
2	0.58 µm	0.34 µm	0.22 µm	73.32 µm	13.60 µm	10.08 µm	45.77 µm
3	0.58 µm	0.27 µm	0.23 µm	89.48 µm	18.43 µm	10.09 µm	49.63 µm
4	0.60 µm	0.46 µm	0.20 µm	19.84 µm	23.58 µm	10.14 µm	10.83 µm
5	0.61 µm	0.43 µm	0.40 µm	12.79 µm	23.97 µm	22.08 µm	11.61 µm
6	0.53 µm	0.17 µm	0.19 µm	81.27 µm	51.14 µm	11.17 µm	11.36 µm
7	0.78 µm	0.17 µm	0.74 µm	17.35 µm	31.71 µm	87.48 µm	33.18 µm
8	0.90 µm	0.15 µm	0.40 µm	95.87 µm	49.08 µm	22.61 µm	66.15 µm
9	0.41 µm	0.41 µm	0.14 µm	38.82 µm	24.22 µm	27.16 µm	15.29 µm
10	0.90 µm	0.15 µm	0.17 µm	95.87 µm	51.86 µm	12.55 µm	30.75 µm

Table 7-4: Design Parameters for Pareto-point samples

To develop the variation model of the Pareto-front, as with PFD/CP, a Monte Carlo simulation was performed to each of the optimal points using foundry variation and mismatch models. 30 samples were chosen for the MC simulation and from these the variation for each performance is calculated. The minimum and maximum range is calculated from the standard deviation and multiplied by 6 for 6-sigma deviation estimation from the mean. Figure 7-26 shows the minimum and maximum plots for VCO phase noise from one of the Monte Carlo samples.

One important aspect that can be seen from the experiment is the sensitivity of the performance functions towards process variations. Figure 7-26 shows a small performance deviation when compared to PFD/CP deviation of figure 7-18. This shows that, between these two circuits PFD/CP noise is more sensitive towards process variations compared to VCO phase noise. Performance sensitivity towards process variation is one of the reasons why it is important to use Pareto-based optimisation method for yield optimisation as explained in chapter 4. The minimum and maximum range of each of the performance functions define the variation model for the VCO and are stored in a data file.

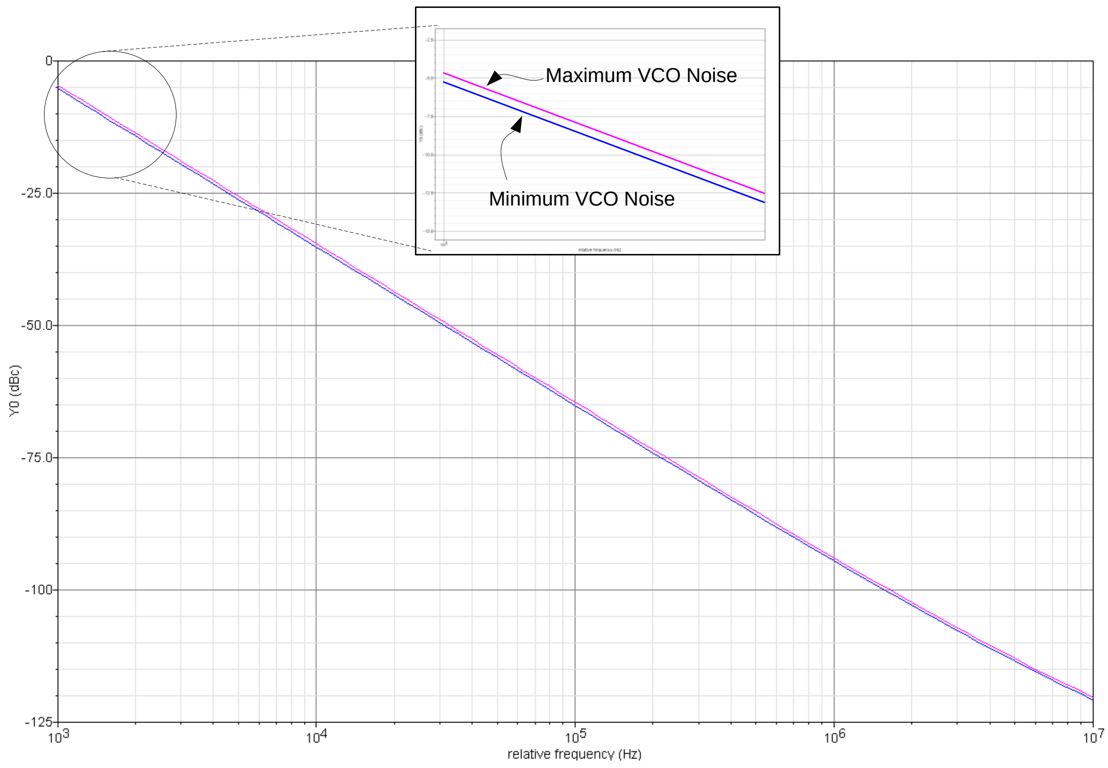


Figure 7-26-: Minimum and maximum plot for VCO phase noise

Both of the data files for performance (nominal) and their variation (minimum and maximum) represent the lookup table for Verilog-A table model function. Table 7-5 shows a selection of sample points from the VCO lookup table. A part of the Verilog-A listing for the VCO table model function for nominal, minimum and maximum performances are shown in figure 7-27 to 7-29. In the listings, Ko is the gain of the VCO and lffl is the VCO phase noise interpolated using the table_model() function from a chosen Ko and VCO current (Ivco). The phase noise of the VCO (lffl) is added to the VCO output using the flicker_noise() function. Similarly, the minimum and maximum performances are determined and interpolated from the variation lookup table.

Design:	**Kvco (Mhz/V):**	**ΔKvco:**	**Jvco (ps):**	**ΔJvco:**	**Ivco (mA) :**	**ΔIvco**
20	997	0.50%	0.13	22%	8.62	2.9%
21	373	0.45%	0.11	22%	3.58	2.7%
22	1090	0.32%	0.29	25%	2.79	2.6%
23	1620	0.30%	0.19	23%	8.46	2.9%
24	2280	0.28%	0.36	26%	4.98	2.7%
27	1850	0.29%	0.21	23%	6.74	2.8%
28	1450	0.29%	0.12	22%	6.16	2.8%
29	1600	0.35%	0.30	25%	2.68	2.6%

Table 7-5: Samples Points from VCO lookup table

```
analog begin
ko = (fmax-fmin)/(vmax-vmin);
lffl = $table_model(ko,Ivco, "vco_data.tbl", "3L,3L");
V(out) <+ laplace_nd(V (in), {fmax-fmin/1},{0,1})
  + flicker_noise(lffl, 3, "VCO_flicker")
  + flicker_noise(lfwh, 2, "VCO_white");
end
endmodule
```

Figure 7-27: VCO table model function for nominal performance

```
analog begin
…
…
//minimum variation for Ivco
Ivco_min = $table_model(Ivco, "Ivcomin_data.tbl", "3L");
$fwrite(file_ptr1, "%e", Ivco_min);

// minimum variation for fmin and fmax
min_fmin = $table_model(fmin, "fmin_mindata.tbl", "3L");
$fwrite(file_ptr2, "%e", min_fmin);
min_fmax = $table_model(fmax, "fmax_mindata.tbl", "3L");
$fwrite(file_ptr3, "%e", min_fmax);

// minimum variation for ko
ko_min = (min_fmax-min_fmin)/(vmax-vmin);
$fwrite(file_ptr4, "%e", ko_min);
ko = (fmax-fmin)/(vmax-vmin);
lffl = $table_model(ko,Ivco, "vco_data.tbl", "3L,3L");

// minimum variation for lffl noise
lffl_min = $table_model(lffl, "lfflmin_data.tbl", "3L");
V(out) <+ laplace_nd(V (in), {(fmax-fmin)/1},{0,1})
  + flicker_noise(lffl_min, 3, "VCO_flicker")
  + flicker_noise(lfwh, 2, "VCO_white");
…
end
endmodule
```

Figure 7-28: VCO table model function for minimum performance

```
analog begin
…
…
// maximum variation for Ivco
Ivco_max = $table_model(Ivco, "Ivcomax_data.tbl", "3L");
$fwrite(file_ptr1, "%e", Ivco_max);

// maximum variation for fmin and fmax
max_fmin = $table_model(fmin, "fmin_maxdata.tbl", "3L");
$fwrite(file_ptr2, "%e", max_fmin);
max_fmax = $table_model(fmax, "fmax_maxdata.tbl", "3L");
$fwrite(file_ptr3, "%e", max_fmax);

// maximum variation for ko
ko_max = (max_fmax-max_fmin)/(vmax-vmin);
$fwrite(file_ptr4, "%e", ko_max);
ko = (fmax-fmin)/(vmax-vmin);
lffl = $table_model(ko,Ivco, "vco_data.tbl", "3L,3L");

// maximum variation for lffl noise
lffl_max = $table_model(lffl, "lfflmax_data.tbl", "3L");
V(out) <+ laplace_nd(V (in), {(fmax-fmin)/1},{0,1})
  + flicker_noise(lffl_max, 3, "VCO_flicker")
  + flicker_noise(lfwh, 2, "VCO_white");
…
…
end
endmodule
```

Figure 7-29: VCO table model function for maximum performance

7.4.3 PLL System Level Design

Once the multi objective bottom up (MUBU) modelling process for performance and variation has been completed, the top down design strategy (TDCD) can be started. At the system level, a behavioural description of the complete system instantiating all sub-blocks component must be developed. All the individual blocks in the system including the PFD, CP, and VCO were behaviourally modelled using Verilog-A language.

The top level PLL system is developed for nominal, minimum and maximum performances. Each of the models correspond to the sub-blocks behavioural model for nominal and their variations performances. Figure 7-30 shows the PLL top level behavioural model for the nominal performance. The minimum and maximum behavioural model are similar except the sub-block instantiation is taken from their minimum and maximum model.

With the system level behavioural model completed, a top-level multi objective optimisation for PLL can be executed. The PLL performance functions are output frequency range, locking time, current consumption, phase margin and total phase noise as shown earlier in table 7-1. The designable parameters for the PLL optimisation are given in table 7-6. As with previous optimisation, the design parameters are constrained within a reasonable range based on the PLL specifications that define the decision space for the optimisation. A spice testbench netlist for the top-level PLL simulation was created and a multi objective optimisation using NSGA-II algorithm was performed on the PLL system in order to locate the optimum solutions that meet the specifications. The simulation results of the top level behavioural model for all performances are used to determine the quality of the solutions against the optimisation requirement. The locking time and phase margin were evaluated analytically during the optimisation. The total PLL phase noise is calculated by superposition of all of the contributing noise sources as explained in section 7.3.2. Figure 7-31 shows an example of the simulation result for PLL phase noise which is shaped by all the noises from PFD/CP and VCO.

```
module PLL_top(ref_in, pll_out);
inout ref_in, pll_out;
electrical ref_in, pll_out;

parameter real Icp = 10e-6 from(0:1.0);
parameter real lfpfl = 0.0 from [0:1.0);

parameter real C_1 = 1.0e-12 from (0:1.0e-3);
parameter real R_2 = 10.0e3 from (0:1M);
parameter real C_2 = 3.0e-12 from (0:1.0e-3);

parameter real fmin = 300e6 from (100e6:80e7); //hertz
parameter real fmax = 500e6 from (200e6:40e8); //hertz
parameter real Ivco = 13.2e-3 from (1e-3:30e-3);
parameter real lfwh = 0.0 from [0:1.0);

parameter real ratio = 1 from (0:inf);

pfd # (.Icp(Icp), .lfpfl(lfpfl))
pfd1(ref_in, divout, filin);
loopfilter # (.C_1(C_1), .R_2(R_2), .C_2(C_2))
loopfilter1(filin, vcoin);
vco # (.fmin(fmin), .fmax(fmax), .Ivco(Ivco), .lfwh(lfwh))
vco1(vcoin, pll_out);
div # (.ratio(ratio))
divider1(pll_out, divout);

endmodule
```

Figure 7-30: PLL top level behavioural model

Design Parameters	Parameter Ranges
Min. Frequency	100MHz – 500MHz
Max. Frequency	1.2GHz – 2 GHz
Charge pump current	10uA – 100uA
VCO current	1mA – 20mA
Resistor, R	1k – 20k
Capacitor, C2	10p – 20p
Capacitor, C1	C2/10

Table 7-6: PLL system designable parameters

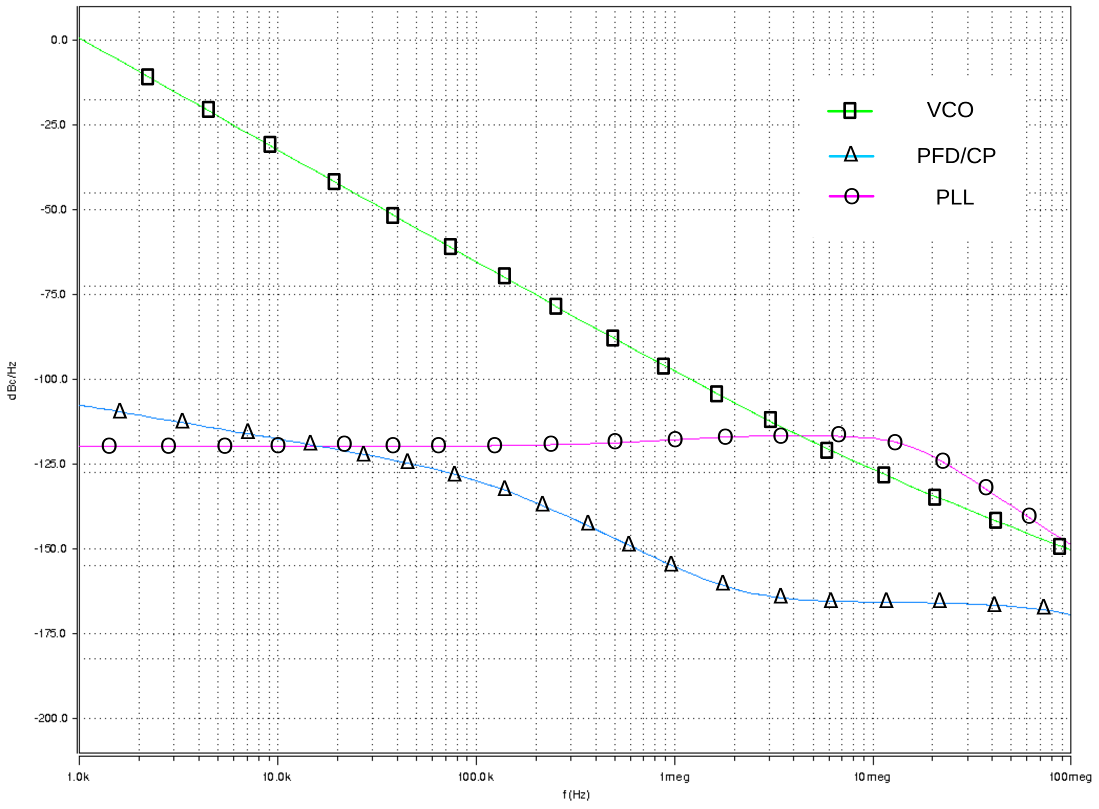


Figure 7-31: Noise simulation result of PLL with all the contributing sources

From the discussion on the PLL performances earlier in this chapter, it can be seen that there are several performance trade-offs occurred in the PLL design. For example, the locking time performance of a PLL system is inversely proportional to its loop bandwidth [107] [100]. This means that, in order for the PLL to lock quickly, the PLL bandwidth must be large. Based on equation 7-8, 7-9 and 7-10, loop bandwidth is directly related to natural frequency and can be determined by VCO gain (K_{VCO}), charge pump current (I_{CP}), C_1 and R. Therefore, the locking time can be reduced by increasing Kvco, I_{CP} C1 and R. However, increasing K_{VCO} will also increase VCO noise hence will increase the total noise of the PLL. Increasing I_{CP} will increase the current consumption hence will influence the total PLL power consumption. Due to this complex trade-off, it is very useful to run multi-objective optimisation and select the best optimal solution from several solution points. In addition to that, with the variation model included in the optimisation, a solution that meets the performance specifications including their variations can be selected.

Table 7-7 shows some samples of the PLL optimal solutions obtained from the multi-objective optimisation including the system minimum and maximum variation. Looking at table 7-7, without looking at the minimum and maximum performances that obtained from the variation model, design no.6 ,7,8,9 and 10 are all solutions that meet the PLL specifications. However, with the variation considered, some of these solutions fail below the specifications. There is only one solution that passes the specifications with variation consideration, that is solution no.9. Therefore, with the help of the variation model developed during MUBU stage, a solution that meets the specifications and at the same time sustain the process variation can be determined. This in turn, will improve overall yield of the PLL system. Figure 7-32 shows the phase noise performances for 3 design points (design point 9, 4 and 10) with the specification boundaries. As can be seen from the figure, design point no. 4 doesn’t meet the phase noise specification for the nominal, minimum and maximum performances and design point no. 10 fails the specification at its maximum variation. Only design point no. 9 meet the specification for nominal and its variations. Therefore, choosing design point no. 9 will meet the specification even when considering the variability.

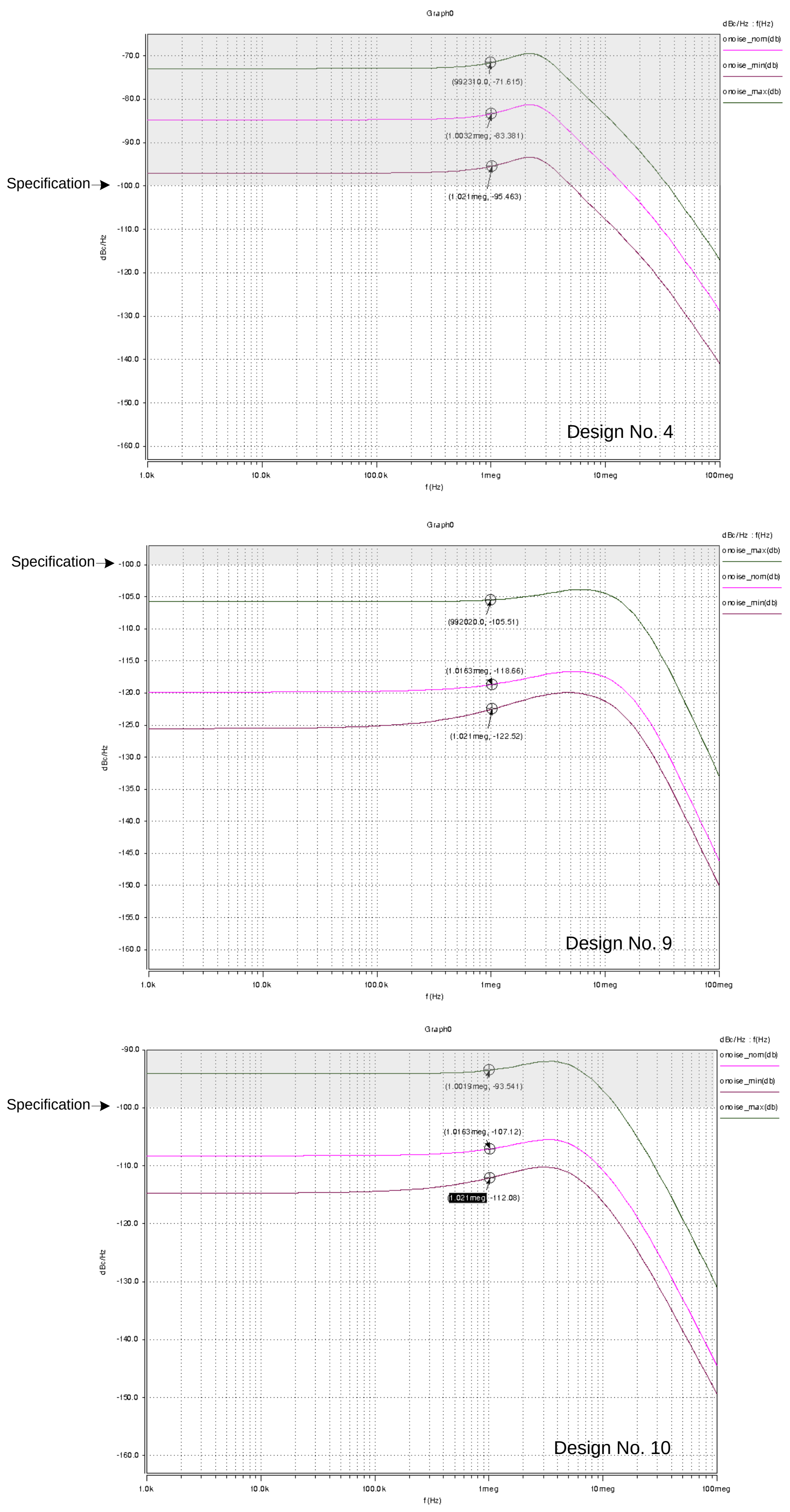


Figure 7-32: Phase noise plots for design point no. 4, 9 and 10

Design Points	Nominal						Minimum						Maximum					
	Noise	**Itot**	**PM**	**Lt**	**fmin**	**fmax**	**Noise**	**Itot**	**PM**	**Lt**	**fmin**	**fmax**	**Noise**	**Itot**	**PM**	**Lt**	**fmin**	**fmax**
1	-68.2	5mA	37.5 deg	330n	470M	2.75G	-87.4	4.5mA	34.2 deg	313n	465.7M	2.45G	-60.1	5.1mA	39.9 deg	353n	475M	3G
2	-108	5.1mA	28 deg	147n	470M	2.75G	-114	4.6mA	25.8 deg	140n	465.7M	2.45G	-94	5.2mA	31.3 deg	158n	475M	3G
3	-84.3	5mA	48.8 deg	233n	470M	2.75G	-96.6	4.5mA	44 deg	211n	465.7M	2.45G	-72.5	5.1mA	51 deg	256n	475M	3G
4	-83.4	7.7mA	47.2 deg	400n	238M	1.4G	-95.5	7.6mA	43 deg	354n	233M	1.1G	-71.5	7.7mA	51.3 deg	440n	241M	1.7G
5	-103	7.7mA	56 deg	281n	238M	1.4G	-108	7.6mA	55 deg	251n	233M	1.1G	-86.2	7.8mA	56.4 deg	311n	241M	1.7G
6	-110	4.1mA	60 deg	235n	407M	1.53G	-114.6	4.05mA	49.9 deg	197n	402M	1.3G	-97	5mA	56.3 deg	272n	410M	1.8G
7	-118	4.1mA	50 deg	181n	407M	1.53G	-121	4.1mA	39.7 deg	153n	402M	1.3G	-105	5mA	56 deg	210n	410M	1.8G
8	-116	2.9mA	50.1 deg	185n	437M	1.52G	-120	2.81mA	40 deg	155n	432M	1.3G	-103	3mA	56 deg	215n	440M	1.83G
9	-119	2.9mA	55.6 deg	491n	437M	1.52G	-123	2.81mA	48.5 deg	424n	432M	1.3G	-106	3mA	55 deg	510n	440M	1.83G
10	-108	2.8mA	54 deg	213n	437M	1.52G	-113	2.7mA	45 deg	179n	432M	1.3G	-94	2.95mA	56.3 deg	250n	440M	1.83G

▨ : Fail below specifications

🟩 : All pass the specifications

Table 7-7: PLL system level optimum samples

Once the best design solution has been selected, the design parameters of this solution will be taken as the specifications for the PLL sub-blocks (i.e. VCO, CP and LF) in order to determine the circuit level design parameters (i.e. transistor size). Table model function of the lower level sub-blocks can be used to determine the circuit sizes. Table 7-8 shows the design parameters for the individual blocks of the PLL system interpolated from the lookup table. Through this complete top down constraint design methodology, the whole PLL circuit has been sized that will give the optimal performances and produces better overall yield.

PLL Block	Design Parameters
Chage pump	Transistor length : 0.12µm
	Transistor Width : 0.35 µm
	Bias current : 100uA
VCO	Lpntrl : 0.47µm
	Wpctrl : 10.00 µm
	Wnctrl : 10.45 µm
	Lpdelay : 0.34 µm
	Wpdelay : 26.02 µm
	Lndelay : 0.15 µm
	Wndelay : 18.15 µm
Loop filter	R1 : 5 kΩ
	C1 : 1.5pF
	C2 : 15pF

Table 7-8: PLL design parameters for individual blocks

There is a possibility that during the top level design, the optimisation process could not find the solution that meets all the specifications. For example, let assume that the phase noise specification for the PLL example is less than -110 dBc/Hz. In this case, all the solutions in table 7-7 fail the phase noise specification at least at one of its variation. If such condition happens, the designer has to decide the solution based on the design priority. Perhaps a weighting parameter can be added to the performances based on the priority and the solution that meets the designer priority can be chosen for the design solution.

A transistor level simulation based on the design parameters from table 7-8 has been carried out for the PLL output frequency range and locking time. Figure 7-33 shows the output frequency range for the PLL system based on the 50 MHz reference frequency. The top plot in figure 7-33 is the reference frequency followed by the output frequency showing 500 MHz signal when the divider ratio is 10 and the last plot shows the output frequency at 1.2GHz when the divider ratio is 24. Figure 7-34 shows locking time plots when the PLL operate at minimum output frequency of 500 MHZ and at maximum output frequency of 1.2GHz. Table 7-9 summarises all the rest of the PLL performances including their minimum and maximum range.

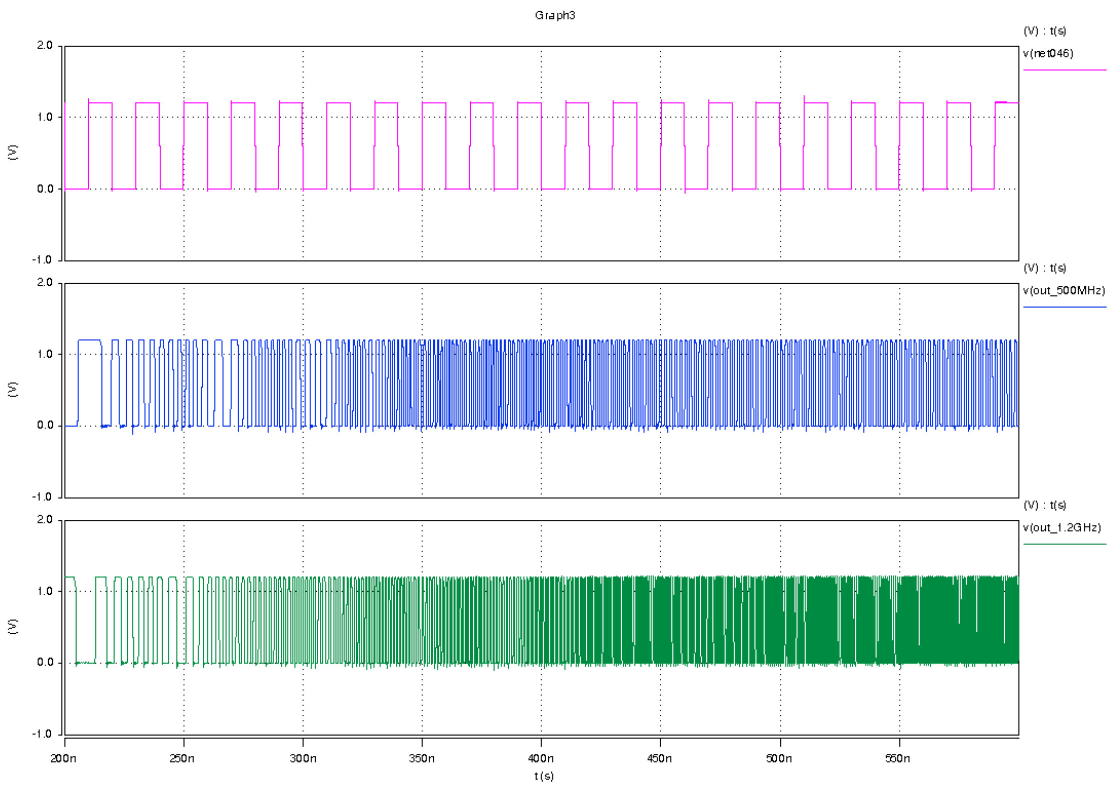


Figure 7-33: PLL output frequency range

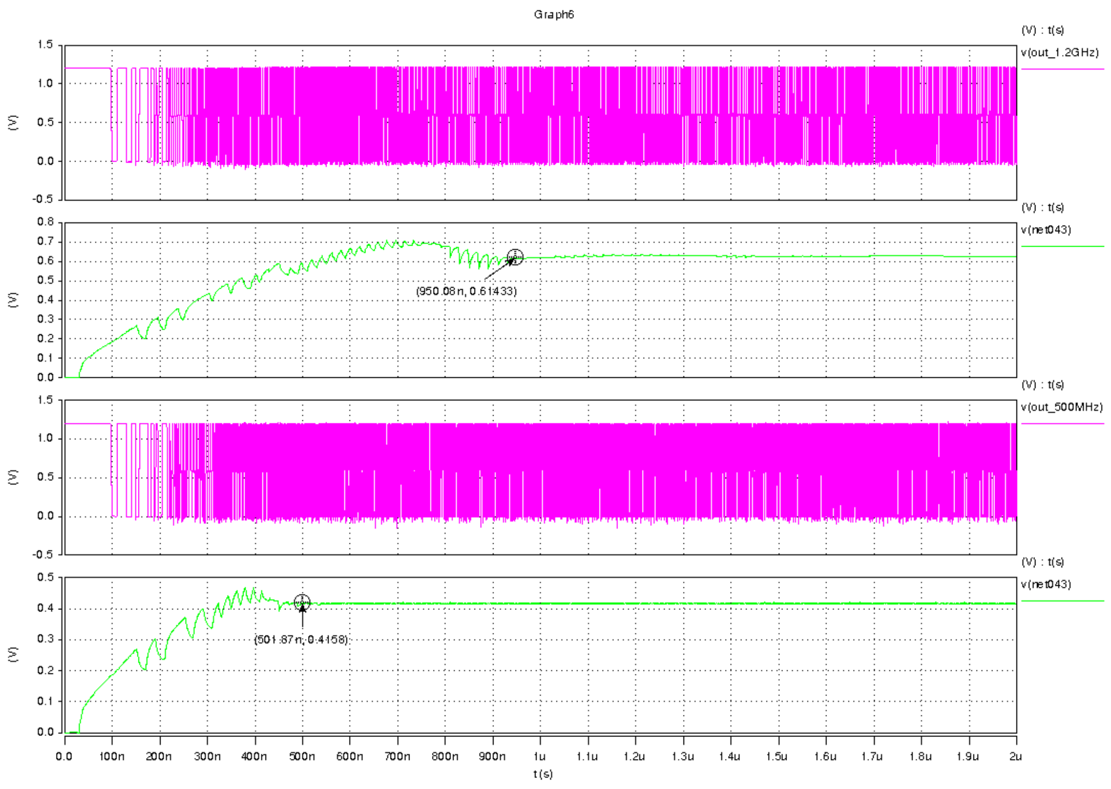


Figure 7-34: PLL locking time for minimum and maximum output frequency

Performance Function	**Specification**	**Nominal Result**	**Minimum Result**	**Maximum Result**
Frequency Range	500MHz - 1.2GHz	437MHz – 1.52GHz	432MHz – 1.30GHz	440MHz – 1.83GHz
Total Current	≤ 5mA	2.9mA	2.81mA	2.9mA
Locking Time	< 1us	502 ns	424 ns	510 ns
Phase Margin	≥ 45 deg	55.6 deg	48.5 deg	55.7 deg
PLL noise	< -100dBc/Hz	-119 dBc/Hz	-123 dBc/Hz	-106 dBc/Hz

Table 7-9: PLL performance results

7.4.4 Design Summary

One of the important aspects of a design methodology for a large system is the computational cost. The decision about design methodology is sometimes a trade off that has to be made between speed and accuracy. As mentioned earlier in this thesis, simulation based design consumes higher simulation time compared to an analytical-based design but produces better accuracy. With recent development in computer technology, the computational overhead is not such a critical factor anymore.

In the PLL system design, the use of behavioural language together with hierarchical optimisation methodologies accelerates the design process. The CPU computational cost employed in the hierarchical-based design is much lower when compared to the `flat' transistor level design and optimisation of a benchmark PLL circuit which requires up to "several weeks or months" [97]. For the proposed method, the higher design time only occurs during the preparation stage where huge number of simulations is needed for the circuit level performance and variation modelling. Table 7-10 summarises the cpu time involved for the complete PLL system design. All the design simulations and optimisations were performed on Ultra Sparc 1.2GHz workstation.

Design Tasks	**CPU Time**
Charge Pump MOO	9 hrs
Charge Pump Monte Carlo	16 hrs
Overall charge pump preparation time	**25 hrs**
Voltage Controlled Oscillator MOO	17 hrs
Voltage Controlled Oscillator MC	25 hrs
Overall VCO preparation time	**42 hrs**
PLL top level MOO	30 minutes
Overall CPU time	**42 hrs 30 minutes**

Table 7-10: PLL system design summary

From table 7-10, it can be seen that, the high CPU time occurred during the preparation stage for the charge pump and VCO modelling.. The variation modelling from the Pareto points can only be started after the multi-objective optimisation for the particular circuit has been completed. Therefore the overall CPU time for the circuit optimisation is the combination of both multi-objective optimisation and Monte Carlo simulation. For example, the overall CPU time for charge pump performance and variation model development is 25 hrs. However, the performance and variation model development for all the individual blocks can be done in parallel, so the CPU cost is determined by the highest contribution which in this example comes from the VCO. The reason for the high simulation time during the MOO is the noise evaluation of the individual blocks that requires a transient noise simulation of the circuit with a small and very well controlled time step. The noise simulation for both of the blocks is the main contributor for the overall simulation time.

Once the preparation stage has been completed, the CPU time required for the PLL design stage through a hierarchical-based optimisation is very fast. From table 7-10, the design time for the PLL system is only 30 minutes. The circuit model developed during the preparation stage can be re-used for other PLL design requirements suggesting a huge time saving can be achieved for the design process.

7.5 Summary

This chapter has demonstrated a complete PLL system level design optimised for performance and yield through a hierarchical-based optimisation methodology. The idea of behavioural performance and variation modelling introduced in chapter 5 and hierarchical optimisation design flow introduced in chapter 6 were used to design the complex performance trade-offs of a PLL system. The PLL system is optimised to meet the performances functions of locking time, phase margin, current consumption, phase noise and output frequency range. The design methodology that integrates both the performance and variation aware analysis, demonstrates its ability to optimise the system level design not only for optimum performances but also for higher yield output. This work shows an example of how the yield can be predicted and optimised from system level point of view.

Chapter 8

Conclusions and Future Work

8.1 Conclusion

A significant portion of the work presented in this thesis has been devoted to the characterization of performance and variation models that can be used for circuit design and optimisation. With reviews of the previous works in this area, simulation-based optimization approach together with Monte Carlo simulation for the variation analysis have been chosen for the circuit design technique. This is due to the accuracy of the proposed methodology that has been given a higher priority for the research work.

The trade-offs among the competing performance objectives were explored using Multi-Objective Optimization (MOO) technique which is based on the Evolutionary Algorithm (EA). This optimization provides a set of solutions on the Pareto front that can be extracted that define a group of solutions to model the performance and variation of a particular circuit. The accuracy of the solution is maintained within the transistor level by incorporating a spice simulator for performance evaluations and Monte Carlo simulation for the variation analysis. In the beginning, the multi-objective optimization used in the algorithm is based on Genetic Algorithm called Weight-Based GA (WBGA). WBGA uses weight vectors that are generated by GA in order to avoid the problem of selecting weight parameters manually. However, with the limitations reported for WBGA in finding solution for non-convex front, a better algorithm called NSGA-II has been used. NSGA-II utilises crowding distance method and several non-dominated sorting procedures to produce a better spreading of Pareto-points which is suitable for more complex circuits. The idea of yield optimisation using multi-objective optimisation approach has been compared with other methods such as design centering and NeoCircuit tool and the results show the benefits gained by the multi-objective optimisation approach.

Simulation-based synthesis creates a good opportunity for modelling. This is due to the huge number of simulation runs that produce a number of data points. The presented research work has successfully built circuit model based on simulation-based optimisation. From the optimal Pareto front which contains a set of trade-off solutions, a lookup table has been constructed that relates all the design parameters to their respective performance functions. For the variation model, a Monte Carlo simulation was performed on the Pareto points and the 6-sigma range was determined for the minimum and maximum points estimation. The variations for each of the Pareto-point solutions are stored in another lookup table. These lookup tables were modelled using table model function of Verilog-A behavioural language. The interpolation method of this function has been used with circuit examples to demonstrate the advantage of the developed model. The results obtained from the circuit simulations show the ability of the model to synthesize a circuit and is comparable with transistor level simulation. A silicon prototype has been produced and the measurement results of the prototype that agree with the simulation data show the ability of the methodology to translate the design into actual product.

In a large system level circuit, the design normally is broken down into smaller sub-block circuits that can be designed and optimised individually. This approach creates several levels of hierarchy. The behavioural performance and variation model is very useful in hierarchical-based system level design. A new hierarchical-based optimisation has been proposed that uses a combination of multi-objective bottom up for performance and variation for sub-block circuits and top-down design for the complete system. The full design flow of the hierarchical-based optimisation has been demonstrated with a 7th order elliptic low pass filter for video application. The results of the optimisation proved that the model can be used to predict and maximised the performances and yield at system level design. In order to demonstrate the application of the proposed methodologies on bigger and complex example, a charge pump PLL has been used as the target application. The higher number of design parameters, complex trade-offs of performance functions and multi domain of circuit analysis including time domain and noise simulation has proved the applicability of the methodologies for a variety of circuit design. The PLL has been designed to meet all the specifications even when process variations are considered. The outcome is a fully sized PLL circuit optimised for performances and yield.

8.2 Accuracy, generality and limitations of the method

The accuracy of the technique has been given a high priority in the presented work. Therefore the approach chosen for the design optimization reflect to this objective. This can be seen in the technique used for the performance optimisation where a simulation-based design and Monte Carlo analysis have been chosen despite the higher computational effort associated with these two techniques.

All these techniques provide better accuracy during the characterisation stage at the circuit level. However at the system level, when a behavioral model is used to simplify the simulation process, accuracy might be limited depending on how close the behavioral model matches the transistor level performance. For example, in chapter 5, the performance of the OTA behavioral model for the filter simulation vary at about 20% from the transistor level simulation. This will affect the accuracy of the system level performances when the model is used at the top level. Therefore, a careful trade-off has to be made during the modelling stage between the accuracy and the complexity of the model. For example, higher number of equations can be added to the behavioral model to improve the accuracy at the expense of the complexity.

The generality of the presented work is highly dependent on the number of performance functions used for the model development. Most of the examples for the OTA performance and variation model development were restricted to two performance functions (Open loop gain and phase margin). In this case, the generality of the model is only limited to the application that related to those performances. The generality of the model can be improved by adding higher number of performance functions. However, the higher the number of performance functions, the higher the number of testbenches and Spice simulations needed that will increase the model development time. This is another trade-off that has to be made at the design stage.

This thesis has presented some ideas that can be used for performance and yield optimization at various hierarchy levels including at the system level design. However, there are still some limitations in the proposed methodology especially when a trade-off has to be made at the design stages. One of the limitations is the accuracy of the method which is highly dependent on the modeling complexity and design cycle time as explained earlier. Computational effort is also another limitation where for a complex multi-domain mixed-signal system, the CPU time for the model development increase significantly. For example, in chapter 7, the complex PLL example has shown a higher preparation time which led to the consideration to reduce some of the optimization parameters such as GA population size, number of GA generation and Monte Carlo simulation samples. This will limit the accuracy of the result that can be achieved by the proposed approach. Therefore, in the future work section (section 8.5), some recommendations have been proposed to mitigate the limitations.

8.3 Project Objectives Achieved

The original hypothesis are reviewed in this section and an assessment of the progress made given for each hypothesis.

- *Hypothesis 1*: Existing yield optimised design methodologies have several inadequacies including in yield modelling and the ability to predict and optimise the yield for system level design.

The first part of this thesis has shown the benefit of considering the process variation parameters in analogue circuit design. A yield optimised design has been shown and a comparison has been made with other techniques such as design centering and a technique using NeoCircuit optimisation tool. The results show the advantage of the multi-objective optimisation technique for yield maximisation and the ability for the method to be used for performance and variation modelling for system level design.

- *Hypothesis 2*: In deep sub-micron technology, where the design complexity and variability has became a great challenge, the accuracy and the ability to translate the simulated results into actual product are very important.

The accurate simulation based optimisation method using multi-objective optimisation and Monte Carlo analysis on the Pareto-points have been used to develop the circuit performance and variation model. The model has been used to design a silicon prototype of 2nd order low pass filter. The measurement results of the prototype and the yield of the prototype samples that agree with the simulation data show the accuracy and efficiency of the method.

- *Hypothesis 3*: Existing approaches for system level design using a hierarchical-based optimisation method do not consider the variations of the sub-block circuits leaving the yield optimisation for the system at the end of the design flow.
- *Hypothesis 4*: A new hierarchical-based optimisation is needed that can incorporate the performance and variation model of analogue circuit in top down system level design flow.

Hypothesis 3 and 4 are closely related and therefore combined. A new hierarchical-based optimisation method that combines multi-objective bottom-up modelling for the sub-block performance and variation parameters and top-down design flow for the complete system design has been developed. With the help of the variation model, the design methodology is capable to optimise the system level design for higher product yield. A 7th order elliptic low pass filter for video application has been designed to demonstrate the methodologies. The performance and yield has been verified with transistor level simulation.

- *Hypothesis 5:* The application of behavioural modelling technology such as Verilog-A allows the integration of various type of systems including mixed-signal and offers huge saving in terms of simulation time.

A complex mixed-signal charge pump PLL system design has been carried out using the proposed methodologies. The behavioural performance and variation model for individual blocks (analogue) in the system has been developed from a multi-objective optimisation result. The top level behavioural simulation instantiating all the sub-blocks is used for the PLL system level optimisation and from this the final design that is optimised for performance and yield is obtained.

Overall, all of the original hypotheses have been addressed.

8.4 Contribution

8.4.1 Specific Contribution

The specific contributions made by this work include:

- Implementation of performance and yield optimisation technique for analogue circuit design using Pareto-based optimisation.
- Development of a combination circuit performance and variation model for analogue circuit design and has been presented at `Design, Automation & Test in Europe (DATE) 2008' conference.
- Development of yield optimisation methodology targeted at system level design using a hierarchical-based optimisation and behavioural performance and variation model.
- System level yield optimisation for Phase Locked Loop (PLL).

8.4.2 Publications

As a direct result of this work, 2 journal papers have beenn published or accepted for publications with a further 1 journal paper submitted for review. 6 papers have also been presented at conferences. The complete list of publications is provided in the `Publication' section of this thesis.

8.5 Future Work

8.5.1 Topological Automation

The methodology proposed in this thesis focuses on circuit sizing stage in analogue synthesis. With the challenges and demand for higher performance circuits, one of the further research area that can be undertaken is to explore automated topology generation for analogue circuit and integrate this technique with the proposed circuit sizing automation to optimise the circuit. In addition to that, the performance and variation modelling technique can be applied to wide variety of analogue circuit topology to create a cell library. With such activity, the performance limitation of a particular topology can be overcome and a better tolerance design solution can be determined.

8.5.2 Hybrid Analytical and Simulation-based Approach

One of the limitations of the simulation-based approach and Monte Carlo simulation is huge computer simulation time. The CPU time consumption of the simulation-based optimisation is directly related to the searching space of the optimisation. The bigger the searching space, the higher the number of simulations required. A useful further work can be undertaken in this area to investigate the ideas to reduce the searching space. A hybrid analytical approach to the simulation-based technique would be a good target. With the analytical approach, circuit equations can be used to add additional constraints to the design parameter so that the decision space is confined to a small area that will give a good result.

8.5.3 Parallel Optimisation

All the examples presented in this thesis were run on a single workstation. As the computing power of PCs increase and the cost of a PC reduced, a parallel optimisation of the circuit modelling can be explored. Simulation-based optimisation has to visit several number of SPICE simulations depending on the number of objective function. On top of that the Monte Carlo simulation need to be done on each of the Pareto-solutions. With the parallel optimisation capability of evolutionary algorithm, the optimisation and Monte Carlo simulations can be distributed to a cluster of workstations and multi-core PCs. For example, 100 Monte Carlo simulations can be reduced to 10 times if the work is distributed to 10 workstations. This will significantly reduce the overall design cycle time for the model development.

Publications

The following are papers published or under review during the course of this thesis work.

[1] Sawal Ali, P.R. Wilson and A.D. Brown, “ Yield predictive model characterization in analogue circuit design”, *IEEE International Symposium on Integrated Circuits 2007, September 2007, Singapore.*

[2] Sawal Ali, R. Wilcock, P.R. Wilson and A.D. Brown, “A new approach for combining yield and performance in behavioural model for analogue integrated circuits”, *Design, Automation and Test in Europe (DATE) 2008, March 3-7, Munich.*

[3] Sawal Ali, R. Wilcock, P.R. Wilson and A.D. Brown, “ Yield model characterisation for analogue integrated circuit using Pareto-optimal surface”, *Cadence Design Network, CDNLive 2008, 28 April, Munich.*

[4] Sawal Ali, R. Wilcock, P.R. Wilson and A.D. Brown, “ Yield model characterisation for analogue integrated circuit using Pareto-optimal surface”, *IEEE International Conference on Electronics, Circuits and Systems, August 2008, Malta.*

[5] Sawal Ali, R. Wilcock and P.R. Wilson, “Behavioural performance and variation modelling for hierarchical-based analogue IC design”, *IEEE International Behavioural Modeling and Simulation Conference (BMAS) September 2008, San Jose, California.*

[6] Sawal Ali, R. Wilcock, P.R. Wilson and A.D. Brown, “Combining yield and performance in behavioural models for analog ICs”, *EDA Tech Forum Journal, Volume 5, Issue 5, December 2008.*

[7] Sawal Ali, R. Wilcock and P.R. Wilson, “Improved performance and variation modelling for hierarchical-based optimisation of analogue integrated circuits”, *Design, Automation and Test in Europe (DATE) 2009, Nice, France.*

[8] Sawal Ali, R. Wilcock and P.R. Wilson, “System level yield optimisation through hierarchical-based design flow”, *IET Electronics Letters, 2009.*

[9] Sawal Ali, R. Wilcock, P.R. Wilson and A.D. Brown, “Performance and variation space modeling using multi-objective optimisation for analogue integrated circuits”, *Submitted to ACM Transactions on Design Automation of Electronic Systems, 2009.*

Reference

[1] G. E. Moore, “Cramming more components onto integrated circuit,” *Electronics*, vol. 38, 1965.

[2] D. L. Wim Kruiskamp, “Darwin: Cmos opamp synthesis by means of a genetic algorithm,” in *Design Automation, 1995. DAC '95. 32nd Conference on*, 1995, pp. 433–438.

[3] A. Ripp, M. Buhler, J. Koehl, J. Bickford, J. Hibbeler, U. Schlichtmann, R. Sommer, and M. Pronath, “Date 2006 special session: Dfm/dfy design for manufacturability and yield - influence of process variations in digital, analog and mixed-signal circuit design,” in *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, vol. 1, 6-10 March 2006, pp. 1–6.

[4] E. Ochotta, R. Rutenbar, and L. Carley, “Synthesis of high-performance analog circuits in astrx/oblx,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 15, no. 3, pp. 273–294, March 1996.

[5] C. L. Mukherjee T. and R. R.A, “Synthesis of manufacturable analog circuits,” in *Computer Aided Design, Nov 1995 (ICCAD) IEEE/ACM International Conference on*, Nov. 1995.

[6] G. Debyser and G. Gielen, “Efficient analog circuit synthesis with simultaneous yield and robustness optimization,” in *Computer-Aided Design, 1998. ICCAD 98. Digest of Technical Papers. 1998 IEEE/ACM International Conference on*, 8-12 Nov 1998, pp. 308–311.

[7] Technical Report, “Neocircuit user's guide,” Cadence Inc., Tech. Rep., 2004.

[8] S. K. Tiwary, P. K. Tiwary, and R. A. Rutenbar, “Generation of yield-aware pareto surfaces for hierarchical circuit design space exploration,” in *Proceedings of the 43rd annual conference on Design automation*. San Francisco, CA, USA: ACM, 2006, pp. 31–36.

[9] User Manual, www.synopsis.com., Synopsys Inc.

[10] Cadence Design Systems Inc., January 2008.

[11] P. Mandal and V. Visvanathan, “Cmos op-amp sizing using a geometric programming formulation,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 20, no. 1, pp. 22–38, Jan. 2001.

[12] L. Carley, G. Gielen, R. Rutenbar, and W. Sansen, “Synthesis tools for mixed-signal ics: progress on frontend and backend strategies,” in *Design Automation Conference Proceedings 1996, 33rd,* 3-7 June 1996, pp. 298–303.

[13] S. R. Hennig, E. and L. Charlack, “An automated approach for sizing complex analogue circuits in a simulation-based flow,” in *Design, Automation and Test in Europe, 2006. DATE '02. Proceedings,* 2002.

[14] S. G. Beenker G., Conway J. and S. A., *Analog CAD for Consumer IC*. Kluwer Academic Publication, 1993.

[15] M. Degrauwe, O. Nys, E. Dijkstra, J. Rijmenants, S. Bitz, B. Goffart, E. Vittoz, S. Cserveny, C. Meixenberger, G. Van der Stappen and H. Oguey, “IDAC: An interactive design tool for analog cmos circuits,” *Solid-State Circuits, IEEE Journal of,* vol. 22, pp. 1106–1116, Dec 1987.

[16] R. Harjani, R. Rutenbar, and L. Carley, “Oasys: a framework for analog circuit synthesis,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 8, no. 12, pp. 1247–1266, Dec. 1989.

[17] F. El-Turky and E. Perry, “Blades: an artificial intelligence approach to analog circuit design,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 8, no. 6, pp. 680–692, June 1989.

[18] A. Fernandez, F.V. Rodriguez-Vazquez, “Symbolic analysis tools - the state-of-the-art,” in *Circuits and Systems, 1996. ISCAS '96. Proceedings of the 1996 IEEE International Symposium on,* 1996.

[19] G. Gielen, H. Walscharts, and W. Sansen, “Analog circuit design optimization based on symbolic simulation and simulated annealing,” *Solid-State Circuits, IEEE Journal of,* vol. 25, no. 3, pp. 707–713, Jun 1990.

[20] G. Gielen, P. Wambacq, and W. Sansen, “Symbolic analysis methods and applications for analog circuits: a tutorial overview,” *Proceedings of the IEEE,* vol. 82, no. 2, pp. 287–304, Feb. 1994.

[21] W. Daems, G. Gielen, and W. Sansen, “Simulation-based generation of posynomial performance models for the sizing of analog integrated circuits,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 22, no. 5, pp. 517–534, May 2003.

[22] H. Koh, C. Sequin, and P. Gray, “Opasyn: a compiler for cmos operational amplifiers,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 9, no. 2, pp. 113–125, Feb. 1990.

[23] E. Hjalmarson, “Studies on design automation of analogue circuit - the design flow,” Ph.D. dissertation, Linkoping University, Institute of Technology, December 2003.

[24] P. Maulik and L. Carley, “Automating analog circuit design using constrained optimization techniques,” in *Computer-Aided Design, 1991. ICCAD-91. Digest of Technical Papers., 1991 IEEE International Conference on*, 11-14 Nov. 1991, pp. 390–393.

[25] M. del Mar Hershenson, S. Boyd, and T. Lee, “Gpcad: a tool for cmos op-amp synthesis,” in *Computer-Aided Design, 1998. ICCAD 98. Digest of Technical Papers. 1998 IEEE/ACM International Conference on*, 8-12 Nov 1998, pp. 296–303.

[26] A. S.-V. William Nye, David C. Riley and A. L.Tits, “Delight.spice: An optimization-based system for the design of integrated circuits,” *IEEE Transactions on Computer-Aided Design*, vol. 7, pp. 501–519, April 1988.

[27] E. S. Ochotta, T. Mukherjee, R. A. Rutenbar, and L. R. Carley, *Practical synthesis of high-performance analog circuits*. Kluwer Academic Publishers, 1998.

[28] F. Medeiro, F. Fernandez, R. Dominguez-Castro, and A. Rodriguez-Vazquez, “A statistical optimization-based approach for automated sizing of analog cells,” in *Computer-Aided Design, 1994., IEEE/ACM International Conference on*, November 6-10, 1994, pp. 594–597.

[29] L. Pillage, X. Huang, and R. Rohrer, “AWEsim: Asymptotic waveform evaluation for timing analysis,” in *Design Automation, 1989. 26th Conference on*, 25-29 June 1989, pp. 634–637.

[30] M. Krasnicki, R. Phelps, R. Rutenbar, and L. Carley, “Maelstrom: efficient simulation-based synthesis for custom analog cells,” in *Design Automation Conference, 1999. Proceedings. 36th*, 21-25 June 1999, pp. 945–950.

[31] R. Phelps, M. Krasnicki, R. Rutenbar, L. Carley, and J. Hellums, “Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 19, no. 6, pp. 703–717, June 2000.

[32] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley Longman, Inc, 1989.

[33] V. Torczon, “On the convergence of the multiderectional search algorithm,” *SIAM , Optimization*, vol. 1, Feb 1991.

[34] A. J.Puhan and T.Tuma, “Analogue integrated circuit sizing with several optimization runs using heuristics for setting initial points,” *Can.J. Electronic Computer Engineering*, vol. 28, July/October 2003.

[35] S. W. H. Box M J, Davies D, “Nonlinear optimization techniques,” *ICI Monograph*, vol. 5, 1969.

[36] G. E. Box, “Evolutionary operation: A method for increasing industrial productivity,” *Applied Statistics*, vol. 6, pp. 81–101, 1957.

[37] J. A. Nelder and R. Mead, “A simplex method for function minimization,” *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[38] M. J. D. Powell, “An efficient method for finding the minimum of a function of several variables without calculating derivatives,” *The Computer Journal*, vol. 7, no. 2, pp. 155–162, Feb. 1964.

[39] M. Al-Saleh and M. Mir, “A modified univariate search algorithm,” in *Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on*, vol. 6, 1999, pp. 306–309 vol.6.

[40] R. Hooke and T. A. Jeeves, “Direct search solution of numerical and statistical problems,” *J. ACM*, vol. 8, no. 2, pp. 212–229, 1961.

[41] J. Bandler, “Optimization methods for computer-aided design,” *Microwave Theory and Techniques, IEEE Transactions on*, vol. 17, no. 8, pp. 533–552, 1969.

[42] J.K. Fidler and R.E. Massara, “Computer optimization of frequency selective networks,” in *Proc European Conference on Circuit Theory and Design IEE Conference Publication*, 1974.

[43] F. Emery, M. O'Hagen, and S. Nolte, “Optimal design of matching networks for microwave transistor amplifiers,” in *MTT International Symposium Digest, 1966*, vol. 66, 1966, pp. 101–107.

[44] D.G. Luenberger, “Linear and Nonlinear Programming”, Addison Wesley, 1984.

[45] C. Desoer and S. Mitra, “Design of lossy ladder filters by digital computer,” *Circuit Theory, IRE Transactions on*, vol. 8, no. 3, pp. 192–201, 1961.

[46] Y. Lam and M. Zwolinski, “Analogue circuit synthesis from performance specifications,” in *Mixed-Signal AHDL/VHDL Modelling and Synthesis, IEE Collloquium on*, 1997.

[47] V. Litovski and M. Zwolinski, *VLSI Circuit Simulation and Optimization*. Chapman & Hall, Dec. 1997.

[48] H. Y. Huang, “Unified approach to quadratically convergent algorithms for function minimization,” *Journal of Optimization Theory and Applications*, vol. 5, no. 6, pp. 405–423, Jun. 1970.

[49] S. Kirkpatrick, C.D. Gelatt and M.P.J. Vecchi, “Optimisation by simulated annealing,” *Science*, pp. 671–680, 1983.

[50] P. Wilson, J. Ross, and A. Brown, “Optimizing the jiles-atherton model of hysteresis by a genetic algorithm,” *Magnetics, IEEE Transactions on*, vol. 37, no. 2, pp. 989–993, March 2001.

[51] K. Deb, *Multi-Objective Optimizatin Using Evolutionary Algorithms*. John Wiley & Sons Ltd, 2001.

[52] L. E. Hajela, P. and C. Lin, “Genetic algorithms in structural topology optimization,” in *NATO Advanced Research Workshop on Topology Design of Structures*, 1993, pp. 117–133.

[53] J. Horn, N. Nafpliotis, and D. Goldberg, “A niched pareto genetic algorithm for multiobjective optimization,” in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, 1994, pp. 82–87 vol.1.

[54] E. Zitzler and L. Thiele, “An evolutionary algorithm for multiobjective optimization: The strength pareto approach,” *Swiss Federal Institute of Technology, TIK-Report*, vol. 43, 1998.

[55] D. Goldberg and K. Deb, “A comparison of selection schemes used in genetic algorithms,” *Foundations of Genetic Algorithms 1 (FOGA-1)*, pp. 69–93, 1991.

[56] Y. Kumar and Chien-In Henry Chen, “Process variation aware transistor sizing for load balance of multiple paths in dynamic cmos for timing optimization,” *Journal of Computer*, vol. 3, pp. 21–28, 2008.

[57] S. Director, P. Feldmann and K. Krishna, “Optimization of parametric yield: A tutorial,” in *Custom Integrated Circuits, 1992, IEEE Conference on.*, 1992.

[58] R. Spence and R. S. Soin, *Tolerance Design Of Electronic Circuits*. Addison Wesley, 1988.

[59] G. Gielen and R.A. Rutenbar, “Computer-aided design of analog and mixed-signal integrated circuits,” *Proceedings of the IEEE*, vol. 88, pp. 1825–1852, 2000.

[60] S. Nassif, A. Strojwas, and S. Director, “A methodology for worst-case analysis of integrated circuits,” *Computer-Aided Design of Integrated Circuits and Systems,IEEE Transactions on*, vol. 5, no. 1, pp. 104–113, 1986.

[61] S. Director and G. Hachtel, “The simplicial approximation approach to design centering,” *Circuits and Systems, IEEE Transactions on*, vol. 24, no. 7, pp. 363–372, Jul 1977.

[62] R. Schwencker, F. Schenkel, H. Graeb, and K. Antreich, “The generalized boundary curve-a common method for automatic nominal design and design centering of analog circuits,” in *Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings*, 27-30 March 2000, pp. 42–47.

[63] J.W. Bandler and H.L. Abdel-Malek, “Optimal centering, tolerancing, and yield determination via updated approximations and cuts,” *Circuits and Systems, IEEE Transactions on*, vol. 10, 1978.

[64] P. Cox, P. Yang, S. Mahant-Shetti, and P. Chatterjee, “Statistical modeling for efficient parametric yield estimation of mos vlsi circuits,” *Solid-State Circuits, IEEE Journal of*, vol. 20, no. 1, pp. 391–398, Feb 1985.

[65] B. J. Guardiani C., Scandolara P. and N. G., “Yield optimization of analog ic's using two-step analytic modeling methods,” *Solid-State Circuits, IEEE Journal of*, vol. 28, pp. 778–783, 1993.

[66] Y. Aoki, H. Masuda, S. Shimada, and S. Sato, “A new design-centering methodology for vlsi device development,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 6, no. 3, pp. 452–461, May 1987.

[67] S. Aftab and M. Styblinski, “IC variability minimization using a new cp and cpk based variability/performance measure,” in *Circuits and Systems, 1994. ISCAS '94., 1994 IEEE International Symposium on*, vol. 1, 30 May-2 June 1994, pp. 149–152vol.1.

[68] M. Keramat and R. Kielbasa, “OPTOMEGA: an environment for analog circuit optimization,” in *Circuits and Systems, 1998. ISCAS '98. Proceedings of the 1998 IEEE International Symposium on*, vol. 6, 31 May-3 June 1998, pp. 122–125 vol.6.

[69] H. Abdel-Malek and A.-K. Hassan, “The ellipsoidal technique for design centering and region approximation,” in *Circuits and Systems, 1989., IEEE International Symposium on*, 8-11 May 1989, pp. 2056–2059 vol.3.

[70] K.Krishna and S. Director, “The linearized performance penalty (LPP) method for optimization of parametric yield and its reliability.” *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 14(12), pp. 1557–1568, December 1995.

[71] F. Leyn, W. Daems, G. Gielen, and W. Sansen, “A behavioral signal path modeling methodology for qualitative insight in and efficient sizing of cmos opamps,” in *Computer-Aided Design, 1997. Digest of Technical Papers., 1997 IEEE/ACM International Conference on*, 9-13 Nov. 1997, pp. 374–381.

[72] G. Gielen, K. Swings, and W. Sansen, “An intelligent design system for analogue integrated circuits,” in *Proceedings of the conference on European design automation*. Glasgow, Scotland: IEEE Computer Society Press, 1990, pp. 169–173.

[73] D. Pescovitz, “1972: The release of spice, still the industry standard tool for integrated circuit design,” Lab Notes: Research from the Berkeley College of Engineering, May/June 2002.

[74] R. E. Massara, *Optimization methods in electronic circuit design*. (Harlow, Essex, England, New York): Longman Scientific & Technical, Wiley, 1991.

[75] W.K. Chen, *The circuits and filters handbook*. CRC Press, 2003.

[76] Dan FitzPatric and Ira Miller, *Analog Behavioural Modeling with The Verilog-A Language*. Kluwer Academic Publishers, 1998.

[77] B. De Smedt and G. Gielen, “Holmes: capturing the yield-optimized design space boundaries of analog and rf integrated circuits,” in *Design, Automation and Test in Europe Conference and Exhibition, 2003*, 2003, pp. 256–261.

[78] H. Liu, A. Singhee, R. Rutenbar, and L. Carley, “Remembrance of circuits past: macromodeling by data mining in large analog design spaces,” in *Design Automation Conference, 2002. Proceedings. 39th*, 10-14 June 2002, pp. 437–442.

[79] T.K. Yu, S. M. Kang, I. Haji, and T. Trick, “Statistical performance modeling and parametric yield estimation of mos vlsi,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 6, no. 6, pp. 1013–1022, November 1987.

[80] G. Wolfe and R. Vemuri, “Extraction and use of neural network models in automated synthesis of operational amplifiers,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 22, no. 2, pp. 198–212, Feb. 2003.

[81] S.K. Tiwary, S. Velu, R.A Rutenbar and T. Mukherjee, “Pareto optimal modeling for efficient pll optimization,” in *Tehcnical Proceeding, 2004 NSTI Nanotechnolgy Conference and Trade Show*, 2004.

[82] V. Bourenkov, K. McCarthy, and A. Mathewson, “Mos table models for circuit simulation,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 3, pp. 352–362, March 2005.

[83] W.Y. Yang, C. Wenmu, C. Tae-sang and J. Morris, *Applied Numerical Methods Using Matlab*. John Wiley & Sons, 2005.

[84] T. Eeckelaert, T. McConaghy, and G. Gielen, "Efficient multiobjective synthesis of analog circuits using hierarchical pareto-optimal performance hypersurfaces," in *Design, Automation and Test in Europe, 2005. Proceedings*, 2005, pp. 1070–1075 Vol. 2.

[85] T. Eeckelaert, R. Schoofs, G. Gielen, M. Steyaert, and W. Sansen, "Hierarchical bottom-up analog optimization methodology validated by a delta–sigma a/d converter design for the 802.11a/b/g standard," in *Proceedings of the 43rd annual conference on Design automation*. San Francisco, CA, USA: ACM, 2006, pp. 25–30.

[86] G. Gielen, T. McConaghy, and T. Eeckelaert, "Performance space modeling for hierarchical synthesis of analog integrated circuits," in *Design Automation Conference, 2005. Proceedings. 42nd*, 13-17 June 2005, pp. 881–886.

[87] H. Chang, A. Sangiovanlli-Vincentelli, F. Balarin, E. Charbon, U. Choudhury, G. Jusuf, E. Liu, E. Malavasi, R. Neff, and P. Gray, "A top-down, constraint-driven design methodology for analog integrated circuits," in *Custom Integrated Circuits Conference, 1992., Proceedings of the IEEE 1992*, May 3-6, 1992, pp. 8.4.1–8.4.6.

[88] H. Chang, E. Charbon, U. Choudhury, A. Demir, E. Felt, E. Liu, E. Malavasi, A. Sangiovanni-Vincentelli, and I. Vassiliou, *A Top-Down, Constraint-Driven Design Methodology for Analog Integrated Circuits*. Norwell, Massachusetts 02061 USA: Springer Verlag, 1999.

[89] S. Donnay, G. Gielen, W. Sansen, W. Kruiskamp, D. Leenaerts, S. Buytaert, K. Marent, M. Buckens, and C. Das, "Using top-down cad tools for mixed analog/digital asics: a practical design case," *Analog Integr. Circuits Signal Process.*, vol. 10, no. 1-2, pp. 101–117, 1996.

[90] R. Iskander, M. Dessouky, M. Aly, M. Magdy, N. Hassan, N. Soliman, and S. Moussa, "Synthesis of cmos analog cells using amigo," in *Proceedings of the conference on Design, Automation and Test in Europe: Designers' Forum - Volume 2*. IEEE Computer Society, 2003.

[91] R. Harjani and J. Shao, "Feasibility and performance region modeling of analog and," *Analog Integrated Circuits and Signal Processing*, vol. 10, pp. 23—43, 1996.

[92] F. D. Bernardinis and M. I. Jordan, "Support vector machines for analog circuit performance representation," *in Proceedings of DAC*, pp. 964—969, 2003.

[93] G. Stehr, H. Graeb and K. Antreich, “Feasibility regions and their significance to the hierarchical optimization of analog and mixed-signal systems,” *International Series of Numerical Mathematics 146*, pp. 167–184, 2003.

[94] I. Bezzam, C. Vinn, and R. Rao, “A fully-integrated continuous-time programmable ccir 601 video filter,” in *Solid-State Circuits Conference, 1995. Digest of Technical Papers. 42nd ISSCC, 1995 IEEE International*, 15-17 Feb. 1995, pp. 296–297,383.

[95] K. Kundert, “Future directions in mixed-signal behavioral modeling,” in *Behavioral Modeling and Simulation, 2002. BMAS 2002. Proceedings of the 2002 IEEE International Workshop on*, 6-8 Oct. 2002, pp. 150–183.

[96] J. Zou, D. Mueller, H. Graeb, and U. Schlichtmann, “Pareto-front computation and automatic sizing of cpplls,” in *Proceedings of the 8th International Symposium on Quality Electronic Design*. IEEE Computer Society Washington, DC, USA, 2007, pp. 481–486.

[97] J. Zou, D. Mueller, H. Graeb and U. Schlichtmann “A cppll hierarchical optimization methodology considering jitter, power and locking time,” in *Proceedings of the 43rd annual conference on Design automation*. San Francisco, CA, USA: ACM, 2006, pp. 19–24.

[98] J. Zou, D. Mueller, H. Graeb, U. Schlichtmann, E. Hennig, and R. Sommer, “Fast automatic sizing of a charge pump phase-locked loop based on behavioral models,” in *Behavioral Modeling and Simulation Workshop, 2005. BMAS 2005. Proceedings of the 2005 IEEE International*, 2005, pp. 100–105.

[99] B. Razavi, *Design of Analog CMOS Integrated Circuits*, 1st ed. McGraw-Hill, Aug. 2000.

[100] R. E. Best, *Phase-locked Loops: Design, Simulation and Applications*. McGraw-Hill Profesional, 2007.

[101] A. Hajimiri, S. Limotyrakis, and T. Lee, “Jitter and phase noise in ring oscillators,” *Solid-State Circuits, IEEE Journal of*, vol. 34, no. 6, pp. 790–804, 1999.

[102] S. W. Wedge, “Pll noise analysis with hspice rf 3rd edition,” Synopsys Inc., Tech. Rep., 2006.

[103] A. Demir and Sangiovanni-Vincentelli, “Simulation and modeling of phase noise in open-loop oscillators,” in *Custom Integrated Circuits Conference, 1996., Proceedings of the IEEE 1996*, 1996, pp. 453–456.

[104] J. Lee, K. Kundert, and B. Razavi, “Modeling of jitter in bang-bang clock and data recovery circuits,” in *Custom Integrated Circuits Conference, 2003. Proceedings of the IEEE 2003*, 21-24 Sept. 2003, pp. 711–714.

[105] T. Lee and A. Hajimiri, “Oscillator phase noise: a tutorial,” *Solid-State Circuits, IEEE Journal of*, vol. 35, no. 3, pp. 326–336, 2000.

[106] Accellera, *Verilog-AMS Language Reference Manual: Analog and Mixed-Signal Extensions to Verilog HDL*, version 2.3 ed., Accellera Organization, August 2008.

[107] P. Hanumolu, M. Brownlee, K. Mayaram, and Un-KuMoon, “Analysis of charge-pump phase-locked loops,” *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 51, no. 9, pp. 1665–1674, 2004.

[108] K.De Jong and W.Spears, “An analysis of the interacting roles of population size and crossover in genetic algorithms,” *Parallel problem solving from nature, Springerlink*, pp. 38-47, 1991.

Appendix A : Spice model Listings

```
M1 N005 N003 0 0 MODN L=2E-6 W=15E-6 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M2 N003 N003 0 0 MODN L=2E-6 W=15E-6 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M3 N002 N002 0 0 MODN L=leff4 W=weff4 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M4 N006 Vin+ N005 N005 MODN L=leff1 W=weff1 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M5 N004 Vin- N005 N005 MODN L=leff1 W=weff1 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M6 Vout N002 0 0 MODN L=leff4 W=weff4 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M7 N001 N006 N006 N001 MODP L=leff2 W=weff2 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M8 N001 N004 N004 N001 MODP L=leff3 W=weff3 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M9 N001 N006 Vout N001 MODP L=leff2 W=weff2 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M10 N001 N004 N002 N001 MODP L=leff3 W=weff3 AD=12.75E-12 AS=12.75E-
12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0

I1 N001 N003 10u
V1 N001 0 3.3V
C2 Vout 0 1p
V2 Vin+ 0 1.6V AC 1mV
R1 Vout Vin- 1000000k
C1 Vin- 0 10u

.param leff1=1.20u
.param leff2=1.92u
.param leff3=0.73u
.param leff4=1.61u

.param weff1=15u
.param weff2=15u
.param weff3=15u
.param weff4=15u

.LIB 'L:\MyFolder\MyPhd\Simulation\spice\hspice\amsc35.lib' NOM

.ac dec 10 1k 100000E+06
.tran 1ns 1us
.measure tran tot_power avg power from=1ns to=1us
.measure ac gain find vdb(vout, vin+) at=1k
.measure ac flat2 find vdb(vout,vin+) at=5G
.measure ac fc when vdb(vout, vin+)=`gain-3.0'
.measure ac unifreq when vdb(vout, vin+)=0
.measure ac phase find vp(vout, vin+) when vdb(vout,vin+)=0
.measure attn PARAM=`gain-flat2'
.OPTIONS PROBE POST MEASOUT
.END
```

Listing A.1: Spice symmetrical-OTA netlist (AC analysis)

```
M1 N006 N004 N002 N002 MODN L=2u W=15E-6 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M2 N004 N004 N002 N002 MODN L=2u W=15E-6 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M3 N003 N003 N002 N002 MODN L=leff4 W=weff4 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M4 N007 Vin+ N006 N006 MODN L=leff1 W=weff1 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M5 N005 0 N006 N006 MODN L=leff1 W=weff1 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M6 Vout N003 N002 N002 MODN L=leff4 W=weff4 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M7 N001 N007 N007 N001 MODP L=leff2 W=weff2 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M8 N001 N005 N005 N001 MODP L=leff3 W=weff3 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M9 N001 N007 Vout N001 MODP L=leff2 W=weff2 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M10 N001 N005 N003 N001 MODP L=leff3 W=weff3 AD=12.75E-12 AS=12.75E-
12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0

I1 N001 N004 10u
V1 N001 0 3.3V
C2 Vout 0 1p
V2 Vin+ 0 0V
V3 0 N002 3.3V

.param leff1=2u
.param leff2=2u
.param leff3=2u
.param leff4=2u

.param weff1=15u
.param weff2=15u
.param weff3=15u
.param weff4=15u

.LIB 'L:\MyFolder\MyPhd\Simulation\spice\hspice\amsc35.lib' NOM

.dc v2 -3v 3v 20mv
.probe v(vout)
.measure dc vos find v(vin+) when v(vout)=0V
.OPTIONS PROBE POST MEASOUT
.END
```

Listing A.2: Spice symmetrical-OTA netlist (voltage offset)

```
M1 N006 N004 N002 N002 MODN L=2u W=15E-6 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M2 N004 N004 N002 N002 MODN L=2u W=15E-6 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M3 N003 N003 N002 N002 MODN L=leff4 W=weff4 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M4 N007 Vin+ N006 N006 MODN L=leff1 W=weff1 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M5 N005 Vout N006 N006 MODN L=leff1 W=weff1 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M6 Vout N003 N002 N002 MODN L=leff4 W=weff4 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M7 N001 N007 N007 N001 MODP L=leff2 W=weff2 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M8 N001 N005 N005 N001 MODP L=leff3 W=weff3 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M9 N001 N007 Vout N001 MODP L=leff2 W=weff2 AD=12.75E-12 AS=12.75E-12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
M10 N001 N005 N003 N001 MODP L=leff3 W=weff3 AD=12.75E-12 AS=12.75E-
12
+PD=16.7E-6 PS=16.7E-6 NRD=33.3333E-3 NRS=33.3333E-3 M=1.0
I1 N001 N004 10u
V1 N001 0 3.3V
C2 Vout 0 1p
V2 Vin+ 0 PWL(0 0 200n 0 201n 3.3 500n 3.3)
V3 0 N002 3.3V

.param leff1=0.35u
.param leff2=1.99u
.param leff3=1.99u
.param leff4=2.00u

.param weff1=15u
.param weff2=15u
.param weff3=15u
.param weff4=15u

.LIB 'L:\MyFolder\MyPhd\Simulation\spice\hspice\amsc35.lib' NOM

.tran 1ns 1us
.print v(vout)
.probe v(vout)
.probe v(vin+)
.measure tran trise trig v(vout) val=0V rise=1 targ v(vout) val=2.8V
+rise=1
.OPTIONS PROBE POST MEASOUT
.END
```

Listing A.3: Spice symmetrical-OTA netlist (Slew Rate)

```
.subckt milota_g1 inm inp out vdd vss
c1 net39 out  0.5e-12
i0 net6 vss  dc=Idc
xm8 net6 net6 vdd vdd   ephsgp_bs3ju w=weff3 l=leff3 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
nbti=0.0
```

```
xm6 net27 net6 vdd vdd   ephsgp_bs3ju w=weff3 l=leff3 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
nbti=0.0
xm5 out net6 vdd vdd   ephsgp_bs3ju w=weff5 l=leff5 nfing=1 ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
nbti=0.0
xm2 net39 inp net27 net27   ephsgp_bs3ju w=weff1 l=leff1 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
nbti=0.0
xm1 net35 inm net27 net27   ephsgp_bs3ju w=weff1 l=leff1 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
nbti=0.0
xm4 out net39 vss vss   enhsgp_bs3ju w=weff4 l=leff4 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
xm3 net35 net35 vss vss   enhsgp_bs3ju w=weff2 l=leff2 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
xm0 net39 net35 vss vss   enhsgp_bs3ju w=weff2 l=leff2 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00

.param Idc=15.3u
.param leff1=0.8u
.param leff2=1.0u
.param leff3=0.13u
.param leff4=0.5u
.param leff5=0.96u
.param weff1=21.5u
.param weff2=46.8u
.param weff3=13.3u
.param weff4=25u
.param weff5=13.3u

.ends milota_g1

xi5 inm inp out net028 net026 milota_g1
c0 out 0  1e-12
c1 inm 0  10e-6
r0 inm out  1e9
v4 inp 0  800e-3 ac 1e-3
v2 net026 0  0.0
v1 net028 0  1.2

.AC DEC        10.0000       1000.00      1000000E+06
.measure ac gain find vdb(out, inp) at=1k
.measure ac phase1 find vp(out, inp) when vdb(out, inp)=0
.probe vdb(out, inp)
.OPTIONS PROBE POST MEASOUT
.LIB 'L:\MyFolder\MyPhd\Simulation\spice\hspice\st12.lib' NOM
.END
```

Listing A.4: Spice Miller-OTA netlist

```
.subckt milota_g1 inm inp out vdd vss
c1 net39 out  0.5e-12
i0 net6 vss  dc=Idc
xm8 net6 net6 vdd vdd   ephsgp_bs3ju w=weff3 l=leff3 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
nbti=0.0
xm6 net27 net6 vdd vdd   ephsgp_bs3ju w=weff3 l=leff3 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
nbti=0.0
xm5 out net6 vdd vdd   ephsgp_bs3ju w=weff5 l=leff5 nfing=1 ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
nbti=0.0
xm2 net39 inp net27 net27   ephsgp_bs3ju w=weff1 l=leff1 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
nbti=0.0
xm1 net35 inm net27 net27   ephsgp_bs3ju w=weff1 l=leff1 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
nbti=0.0
xm4 out net39 vss vss   enhsgp_bs3ju w=weff4 l=leff4 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
xm3 net35 net35 vss vss   enhsgp_bs3ju w=weff2 l=leff2 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
xm0 net39 net35 vss vss   enhsgp_bs3ju w=weff2 l=leff2 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00

.param Idc=18.3u
.param leff1=0.9u
.param leff2=0.9u
.param leff3=1.0u
.param leff4=0.30u
.param leff5=0.68u
.param weff1=15.0u
.param weff2=10.0u
.param weff3=10.7u
.param weff4=28u
.param weff5=40.0u

.ends milota_g1

.param capeff1=2p
.param capeff2=0.34p
.param capeff3=1p

c2 out 0  capeff3
c1 out in  capeff2
c0 net20 0  capeff1
v2 net7 0  1.2
v1 net21 0  1.2
v0 in 0  600e-3 ac 1e-3
xi2 out net20 out net7 0 milota_g1
xi0 out in net20 net21 0 milota_g1

.ac dec 10 1k 1000000E+06
.measure ac gain find vdb(out, in) at=1k
```

```
.measure ac fc when vdb(out, in)=`gain-3.0'
.measure ac gainpeak max vdb(out, in)
.measure ac pbripp PARAM=`gainpeak-gain'
.measure ac minpoint min vdb(out, in)
.measure ac freqmin when vdb(out, in)=`minpoint'
.measure ac point1 when vdb(out, in)=`minpoint+2'
.measure ac peak2 max vdb(out, in) FROM=`point1' TO=500Meg
.measure ac attn PARAM=`gain-peak2'
.measure ac fs when vdb(out, in)=`peak2' fall=1
.measure ac steep PARAM=`fs-fc'
.measure ac fs1 when vdb(out, in) = `-40'
.measure ac fs2 when vdb(out, in) = `-60'
.LIB 'L:\MyFolder\MyPhd\Simulation\spice\hspice\st12.lib' NOM
.END
```

Listing A.5: Spice 2nd order low pass filter netlist

```
.hdl 'otasimple.va'
V11 net047 0 ac=10e-3
C10 net0110 0 c=cap10
C6 net0117 net083 c=cap6
C5 net0117 0 c=cap5
C4 net0121 0 c=cap4
C8 net0118 0 c=cap8
C9 net0116 0 c=cap9
C2 net083 net047 c=cap2
C3 net083 0 c=cap3
C1 net085 0 c=cap10
C7 net0116 net0117 c=cap7
X7 net0110 net0116 net0110 ota ce=-126f gm=gm_ota gm3=117.3u
+cout=-150f ro=106.2k cgd1=-15f cgd2=-15f
X6 net0116 net0118 net0116 ota ce=-126f gm=gm_ota gm3=117.3u
+cout=-150f ro=106.2k cgd1=-15f cgd2=-15f
X5 net0118 net0117 net0116 ota ce=-126f gm=gm_ota gm3=117.3u
+cout=-150f ro=106.2k cgd1=-15f cgd2=-15f
X2 net083 net085 net083 ota ce=-126f gm=gm_ota gm3=117.3u
+cout=-150f ro=106.2k cgd1=-15f cgd2=-15f
X1 net085 net047 net083 ota ce=-126f gm=gm_ota gm3=117.3u
+cout=-150f ro=106.2k cgd1=-15f cgd2=-15f
X4 net0117 net0121 net0117 ota ce=-126f gm=gm_ota gm3=117.3u
+cout=-150f ro=106.2k cgd1=-15f cgd2=-15f
X3 net0121 net083 net0117 ota ce=-126f gm=gm_ota gm3=117.3u
+cout=-150f ro=106.2k cgd1=-15f cgd2=-15f

.param gm_ota=113.8u
.param cap1=2.00p
.param cap2=1.36p
.param cap3=8.27p
.param cap4=1.31p
.param cap5=5.07p
.param cap6=2.00p
.param cap7=2.78p
.param cap8=1.76p
.param cap9=6.81p
.param cap10=1.88p

.ac dec 10 1k 1000000E+06
.measure ac gain find vdb(net0110, net047) at=1k
.measure ac fp when vdb(net0110, net047)=`gain-3.0'
.measure ac minpoint min vdb(net0110, net047) FROM=`gain-3' to=30Meg
```

```
.measure ac freqmin when vdb(net0110, net047)=`minpoint'
.measure ac peak2 max vdb(net0110, net047) FROM=`freqmin' TO=5000Meg
.measure ac attn PARAM=`peak2'
.measure ac fs when vdb(net0110, net047)=`peak2' fall=1
.probe vdb(net0110, net047)
.end
```

Listing A.6: Spice 7th order low pass filter Netlist

```
.subckt ota_g1 inm inp out
i0 net20 VSS!  dc=10e-6
xm3 net23 net23 VDD! VDD!  ephsgp_bs3ju w=weff2 l=leff2 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
nbti=0.0
+lpe=0
xm2 out net23 VDD! VDD! ephsgp_bs3ju w=weff2 l=leff2 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
nbti=0.0
+lpe=0
xm1 out inm net20 VSS!   enhsgp_bs3ju w=weff0 l=leff0 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
lpe=0
xm0 net23 inp net20 VSS!   enhsgp_bs3ju w=weff0 l=leff0 nfing=1
ncrsd=1.0
+number=1.0 srcefirst=1 ngcon=1 mismatch=1 po2act=-1.00000000e+00
lpe=0

.param leff0=1u
.param leff2=1u
.param weff0=10u
.param weff2=10u

.ends ota_g1

XI15 VINM VINP VOUT ota_g1
C0 VOUT 0  1E-12
V4 VINP 0  600E-3 AC 1E-3
V5 VINM 0  600E-3
V3 0 VSS!  0.0
V0 VDD! 0  1.2

.AC DEC        10.0000       1000.00      1000E+06
.measure ac gain find vdb(VOUT, VINP) at=1k
.measure ac phase find vp(VOUT, VINP) when vdb(VOUT, VINP)=0
.probe vdb(vout, vin+)
.OPTIONS PROBE POST MEASOUT

.LIB 'L:\MyFolder\MyPhd\Simulation\spice\hspice\st12.lib' NOM
.END
```

Listing A.7: Spice single stage OTA Netlist

```
.param vctrl=0.4
C0 vcoout 0 1a
vctrl net2 0 DC vctrl
V1 vss 0 0
V0 vdd 0 1.2
X1 net2 vdd vcoout vss vco

.subckt vco vctrl vdd vout vss
XM21 vout net12 net28 vss ENHSGP_BS3JU w=wndelay l=lndelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
XM20 net12 net16 net32 vss ENHSGP_BS3JU w=wndelay l=lndelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
XM19 net16 net20 net36 vss ENHSGP_BS3JU w=wndelay l=lndelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
XM18 net20 net24 net40 vss ENHSGP_BS3JU w=wndelay l=lndelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
XM17 net24 vout net44 vss ENHSGP_BS3JU w=wndelay l=lndelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
XM16 net28 vctrl vss vss ENHSGP_BS3JU w=wndelay l=lndelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
XM15 net32 vctrl vss vss ENHSGP_BS3JU w=wndelay l=lndelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
XM14 net36 vctrl vss vss ENHSGP_BS3JU w=wndelay l=lndelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
XM13 net40 vctrl vss vss ENHSGP_BS3JU w=wndelay l=lndelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
XM7 net48 vctrl vss vss ENHSGP_BS3JU w=wnctrl l=lpnctrl nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
XM1 net44 vctrl vss vss ENHSGP_BS3JU w=wndelay l=lndelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
XM12 vout net12 net54 vdd EPHSGP_BS3JU w=wpdelay l=lpdelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
+lpe=0
XM11 net12 net16 net58 vdd EPHSGP_BS3JU w=wpdelay l=lpdelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
+lpe=0
XM10 net16 net20 net62 vdd EPHSGP_BS3JU w=wpdelay l=lpdelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
+lpe=0
XM9 net20 net24 net66 vdd EPHSGP_BS3JU w=wpdelay l=lpdelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
+lpe=0
XM8 net24 vout net70 vdd EPHSGP_BS3JU w=wpdelay l=lpdelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
+lpe=0
XM6 net54 net48 vdd vdd EPHSGP_BS3JU w=wpdelay l=lpdelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
+lpe=0
XM5 net58 net48 vdd vdd EPHSGP_BS3JU w=wpdelay l=lpdelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
+lpe=0
XM4 net62 net48 vdd vdd EPHSGP_BS3JU w=wpdelay l=lpdelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
+lpe=0
XM3 net66 net48 vdd vdd EPHSGP_BS3JU w=wpdelay l=lpdelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
+lpe=0
XM2 net70 net48 vdd vdd EPHSGP_BS3JU w=wpdelay l=lpdelay nfing=1
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
+lpe=0
XM22 net48 net48 vdd vdd EPHSGP_BS3JU w=wpctrl l=lpnctrl nfing=1
```

```
+ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
+lpe=0

.param lpnctrl=1u
.param lpdelay=1u
.param lndelay=1u

.param wpctrl=171u
.param wpdelay=1u
.param wnctrl=57u
.param wndelay=1u

.ends vco

.OPTIONS PROBE POST
.options HBTRANINIT=100n
.options HBTRANPTS=20
.options HBCONTINUE=0
.options phnoise_lorentz=0

.sweepblock vtune_sweep
+ 0.4 1.2 0.2

.IC v(vcoout)=1V
.HBOSC tones=1200Meg nharms=12
+ probenode= vcoout,vss 0.6
+ sweep vctrl sweepblock=vtune_sweep

*-------------------------------------------------
* for plotting HB transient wavform of v(vcoout)
* The output file is ~.hr0
*-------------------------------------------------
.probe hbtran v(vcoout)

*---------------------------------------------------------------------
* for plotting HB oscillation spectrum of v(vcoout) and i(v0) for the
current
* convert to time domain yield a transient waveform similar as hbtran
* The output file is ~.hb0
*---------------------------------------------------------------------
.probe HBOSC v(vcoout)
.probe hbosc i(v0) [0]

*-----------------------------------------------------------
* for ploting harmonics frequency. the output file is ~.hb0
* with voltage control sweep, VCO gain can be determined
*-----------------------------------------------------------
.probe HB hertz[1]

*----------------------
* Phase Noise Analysis
*----------------------
.phasenoise V(vcoout,vss) dec 10 1k 1e7

*------------------------------------------
* for plotting phase noise again frequency
* output file is ~.pn0
*-----------------------------------------
.probe phasenoise phnoise

*-----------------------------------------------
```

```
* for plotting jitter from phase noise analysis
* output file is ~.jt0
*---------------------------------------
.probe phasenoise phnoise jitter

*---------------------------------------------------------
* for measuring RMS period jitter from phase noise result
*---------------------------------------------------------
* period jitter measurement use the full offset frequency
* sweep range given in the phase noise analysis. The
* from and to parameters are ignored.
*---------------------------------------------------------
.MEASURE PHASENOISE rjper PERJITTER phnoise from 1k to 10Meg
*.measure phasenoise rjper2 perjitter phnoise when v(vctrl)=0.2

*-----------------------------------------------
* To measure VCO Gain, Kvco
* Vmax-Vmin = 1.2V - 0.2V = 1.0V
*-----------------------------------------------
.measure hb freqmin min PAR(HERTZ[1]);1_Mag
.measure hb freqmax max PAR(HERTZ[1]);1_Mag
.measure hb deltafreq PARAM=`freqmax-freqmin'
.measure hb kvco PARAM=`deltafreq/1.0'

.measure phasenoise pn_freqmin find phnoise at 1k

*---------------------------------------------------
* To measure total maximum current and maximum power
* at DC frequency. Measured at power supply
*---------------------------------------------------
.measure hbosc totcurr max i(v0) [0]
.measure hbosc totpwr max p(v0) [0]

*----------------------------------------------
* ST0.12um Models file for simulator hspiceS
*----------------------------------------------
.lib '/home/sawal/phd/modelfile/st12/common_poly.lib' PRO_TT
.lib '/home/sawal/phd/modelfile/st12/common_active.lib' PRO_TT
.lib '/home/sawal/phd/modelfile/st12/common_go1.lib' PRO_TT
.lib '/home/sawal/phd/modelfile/st12/common_go2.lib' PRO_TT
.lib '/home/sawal/phd/modelfile/st12/mos_bsim3_HS.lib' moshs_TT
.lib '/home/sawal/phd/modelfile/st12/mos_bsim3_LL.lib' mosll_TT
.lib '/home/sawal/phd/modelfile/st12/mos_bsim3_3V3.lib' mos3v3_TT

.END
```

Listing A.8: Spice VCO netlist

```
.OPTIONS PROBE POST MEASOUT

R0 out net018 2K
C1 net018 0 10p
C0 out 0 1p
V2 net14 0 1.2
V1 net7 0 pulse 1.2 0.0 0ns 1fs 1fs 1n 2n
V0 net5 0 pulse 1.2 0.0 0ns 1fs 1fs 1n 2n
XI1 net19 net18 out net21 net20 net14 0 cp_1
XI0 net19 net18 net7 net5 net21 net20 pfd
```

```
.subckt cp_1 dw dwb out up upb vdd vss
    I2 vdd net045 dc=100u
    XM7 net049 net045 vss vss ENHSGP_BS3JU w=0.15e-6 l=0.13e-6
nfing=1 ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM10 net045 net045 vss vss ENHSGP_BS3JU w=0.15e-6 l=0.13e-6
nfing=1 ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM8 net20 net045 vss vss ENHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM5 out dw net20 vss ENHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM4 out dwb out vss ENHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM3 vdd dwb net20 vss ENHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
    +ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM6 net33 net049 vdd vdd EPHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0 lpe=0
    XM9 net049 net049 vdd vdd EPHSGP_BS3JU w=0.15e-6 l=0.13e-6
nfing=1 ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0 lpe=0
    XM2 out up out vdd EPHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0 lpe=0
    XM1 out upb net33 vdd EPHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0 lpe=0
    XM0 vss up net33 vdd EPHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0 lpe=0
.ends cp_1

.subckt inv_gate in out
    V0 net12 0 1.2
    XM1 out in 0 0 ENHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1 ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM0 out in net12 net12 EPHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
    +ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
    +lpe=0
.ends inv_gate

.subckt and_gate A B out
    V0 net4 0 1.2
    XM5 out net28 0 0 ENHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM4 net9 B 0 0 ENHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1 ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM3 net28 A net9 0 ENHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM2 out net28 net4 net4 EPHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
    +ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
    +lpe=0
    XM1 net28 A net4 net4 EPHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
    +ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
    +lpe=0
    XM0 net28 B net4 net4 EPHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
    +ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
    +lpe=0
.ends and_gate

.subckt dff_1 D Q Res clk
    XM7 Q net16 0 0 ENHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1 ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM6 net12 net20 0 0 ENHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM5 net16 clk net12 0 ENHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
    +ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM4 net20 Res 0 0 ENHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 lpe=0
    XM3 Q net16 D D EPHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1 ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0 lpe=0
    XM2 net16 net20 D D) EPHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0 lpe=0
    XM1 net20 Res net30 D EPHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
    +ncrsd=1 number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0
    +lpe=0
    XM0 net30 clk D D EPHSGP_BS3JU w=0.15e-6 l=0.13e-6 nfing=1
ncrsd=1
    +number=1 srcefirst=1 ngcon=1 mismatch=1 po2act=-1 nbti=0 lpe=0
.ends dff_1

.subckt pfd dw dwb fback fref up upb
    XI8 net7 net044 inv_gate
    XI9 net044 up inv_gate
    XI10 net7 upb inv_gate
    XI11 net3 dwb inv_gate
    XI12 net3 net036 inv_gate
    XI13 net036 dw inv_gate
    V0 vdd 0 1.2
    XI3 net7 net3 net8 and_gate
    XI1 vdd net3 net8 fback dff_1
    XI0 vdd net7 net8 fref dff_1
.ends pfd

.SN tone=500MEG nharms=10 trinit=100n
.SNNOISE V(out) V1
+DEC 20 1k 100MEG
+ [0,1]


.PRINT ACPHASENOISE PHNOISE JITTER
.PROBE ACPHASENOISE PHNOISE JITTER
.PROBE SN V(out)
.PROBE SNNOISE onoise


.lib '/home/sawal/phd/modelfile/st12/common_poly.lib' PRO_TT
.lib '/home/sawal/phd/modelfile/st12/common_active.lib' PRO_TT
.lib '/home/sawal/phd/modelfile/st12/common_go1.lib' PRO_TT
.lib '/home/sawal/phd/modelfile/st12/common_go2.lib' PRO_TT
.lib '/home/sawal/phd/modelfile/st12/mos_bsim3_HS.lib' moshs_TT
.lib '/home/sawal/phd/modelfile/st12/mos_bsim3_LL.lib' mosll_TT
```

```
.lib '/home/sawal/phd/modelfile/st12/mos_bsim3_3V3.lib' mos3v3_TT


.END
```

Listing A.9: Spice PFD/CP Netlist

```
.option post
.hdl 'lf.va'
.hdl 'vco.va'
.hdl 'pfd.va'
.hdl 'divider.va'
.hdl 'pll_top.va'

V1 ref 0 pulse 1.2 0.0 0ns 1fs 1fs 900p 1.8n AC=1mV

xa1 ref pll_out pll_top
+ Icp=pfd_current
+ lfpfl=0.0
+ C_1=cap1
+ R_2=res
+ C_2=cap2
+ fmin=min_freq
+ fmax=max_freq
+ Ivco=vco_current
+ lfwh=0.0
+ ratio=divide_by

.PARAM pfd_current=100e-6
.PARAM min_freq=437e6
.PARAM max_freq=1.52e9
.PARAM vco_current=2.79e-3
.PARAM divide_by=1
.PARAM res=5k
.PARAM cap2=10p
.PARAM cap1=cap2/10

.ac dec 20 1k 100Meg
.probe ac vdb(pll_out, ref)
.noise v(pll_out) v1
.print ac vdb(pll_out) onoise onoise(dB)
.probe ac vdb(pll_out) onoise onoise(dB)
.measure ac MSjitter integral `2.0*onoise*onoise'
+ from=1k to=100Meg

.measure vco_gain param = '(max_freq-min_freq)/(1.2-0.2)'
.measure RMSjitter param='sqrt(MSjitter)'
.measure wn
+param='((pfd_current*vco_gain)/(2*3.14*divide_by*cap2))^(1/2)'
.measure wz param = '1/(res*cap2)'
.measure cap_series param = '(cap1*cap2)/(cap1+cap2)'
.measure wp param = '1/(res*cap_series)'
.measure damp_factor param='(Wn*res*cap2)/2'
.measure loop_bwidth
+param='Wn*((1+2*damp_factor^2+((2+4*damp_factor^2+4*damp_factor^4)^1
/2))^1/2)'
.measure lock_time param = '(2*3.14)/Wn'
.measure gain_pfd param = 'pfd_current/(2*3.14)'
.measure wugb param = '(gain_pfd*vco_gain*res)/divide_by'
```

```
.measure test1 param = 'wugb/wz'
.measure test2 param = 'wugb/wp'
.measure atan1 param = 'atan(test1)'
.measure atan2 param = 'atan(test2)'
.measure phase_margin param = 'atan(wugb/wz)-atan(wugb/wp)'
.measure PM_degrees param = 'phase_margin*(180/3.14)'
.measure tot_current param = 'pfd_current + vco_current'

.end
```

Listing A.10: Spice PLL Netlist

Appendix B: Algorithm model Listings

```
/* This is a Multi-Objective GA program.
*********************************************************************
*
*  This program is the implementation of the NSGA-2 proposed by
*
*
*
*  Prof. Kalyanmoy Deb and his students .
*
*
*
*  copyright Kalyanmoy Deb
*********************************************************************
*

18.08.2003: The keepaliven.h file is modified to have normalized
            crowding distance calculation. The previous version of
            the code did not have this feature. This way, maintaining
            a good distribution of solutions in problems having quite
            a different range of objective functions were difficult.
            Hopefully, with this modification, such difficulties will
            not appear. --  K. Deb
18.08.2003: Also the dfit.h file is deleted. It was not needed any
way.

The user have to give the input manualy or through a data file.

The user needs to enter objective functions in func-con.h
The code can also take care of the constraints. Enter the constraints
in the space provided in the func-con.h file.
Constraints must be of the following type:
g(x) >= 0.0
Also normalize all constraints (see the example problem in func-
con.h)


Compilation procedure:  gcc nsga2.c -lm
Run ./a.out with or without an input file

Input data files: Three files are included, but at one time one is
needed
depending on the type of variables used:
inp-r (template file input-real)  : All variables are real-coded
inp-b (template file input-binary): All variables are binary-coded
inp-rb(template file input-rl+bin): Some variables are real and some
are binary
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <fstream.h>

#define square(x) ((x)*(x))
```

```
#define maxpop   500  /*Max population */
#define maxchrom 200  /*Max chromosome length*/
#define maxvar    20  /*Max no. of variables*/
#define maxfun    10  /*Max no. of functions */
#define maxcons   20  /*Max no. of Constraints*/


int gener,       /*No of generations*/
  nvar,nchrom,   /*No of variables*/
  ncons,         /*No of Constraints*/
  vlen[maxvar],  /*Array to store no of bits for each variable*/
  nmut,          /* No of Mutations */
  ncross,        /*No of crossovers*/
  ans;
float seed,      /*Random Seed*/
  pcross,        /*Cross-over Probability*/
  pmut_b, pmut_r,/*Mutation Probability*/
  lim_b[maxvar][2], lim_r[maxvar][2];/*Limits of variable in array*/
float di,        /*Distribution Index for the Cross-over*/
  dim,           /*Distribution Index for the Mutation*/
  delta_fit,     /* variables required forfitness for fitness sharing
*/
  min_fit,
  front_ratio;
int optype,      /*Cross-over type*/
  nfunc,         /*No of functions*/
  sharespace;    /*Sharing space (either parameter or fitness)*/

double coef[maxvar]; /*Variable used for decoding*/

static int popsize,  /*Population Size*/
  chrom;             /*Chromosome size*/

typedef struct       /*individual properties*/
{

  int genes[maxchrom], /*bianry chromosome*/
    rank,              /*Rank of the individual*/
    flag;              /*Flag for ranking*/
  float xreal[maxvar], /*list of real variables*/
    xbin[maxvar];      /*list of decoded value of the chromosome */
  float fitness[maxfun],/*Fitness values */
    constr[maxcons],     /*Constraints values*/
    cub_len,             /*crowding distance of the individual*/
    error;              /* overall constraint violation for the
individual*/
}individual;        /*Structure defining individual*/


typedef struct
{
  int maxrank;            /*Maximum rank present in the population*/
  float rankrat[maxpop];  /*Rank Ratio*/
  int rankno[maxpop];     /*Individual at different ranks*/
  individual ind[maxpop], /*Different Individuals*/
    *ind_ptr;
}population ;             /*Popuation Structure*/

#include "random.h"       /*Random Number Generator*/

#include "input.h"        /*File Takes Input from user*/
```

```
#include "realinit.h"     /*Random Initialization of the populaiton*/
#include "init.h"         /*Random Initialization of the population*/
#include "decode.h"       /*File decoding the binary dtrings*/
#include "ranking.h"      /*File Creating the Pareto Fronts*/
#include "rancon.h"       /*File Creating the Pareto Fronts when
                            Constraints are specified*/
#include "func-con.h"     /*File Having the Function*/
#include "select.h"       /*File for Tournament Selection*/
#include "crossover.h"    /*Binary Cross-over*/
#include "uniformxr.h"    /*Uniform Cross-over*/
#include "realcross2.h"   /*Real Cross-over*/
#include "mut.h"          /*Binary Mutation*/
#include "realmut1.h"     /*Real Mutation*/
#include "keepaliven.h"   /*File For Elitism and Sharing Scheme*/
#include "report.h"       /*Printing the report*/

population oldpop,
  newpop,
  matepop,
  *old_pop_ptr,
  *new_pop_ptr,
  *mate_pop_ptr;
/*Defining the population Structures*/

main()
{
  /*Some Local variables to this Problem (Counters And some other
pointers*/

  int i,j,l,f,maxrank1;
  float *ptr,tot;
  FILE
    *rep_ptr,
    *gen_ptr,
    *rep2_ptr,
    *end_ptr,
    *g_var,
    *lastit;
      //*param_ptr,    // parameter file
  /*File Pointers*/

  //param_ptr = fopen("param.txt", "w");   // parameter file
  rep_ptr = fopen("output.out","w");
  gen_ptr =fopen("all_fitness.out","w");
  rep2_ptr = fopen("ranks.out","w");
  end_ptr = fopen("final_fitness.out","w");
  g_var = fopen("final_var.out","w");
  lastit = fopen("plot.out","w");
  /*Opening the files*/

  old_pop_ptr = &(oldpop);

  nmut = 0;
  ncross = 0;

  /*Get the input from the file input.h*/
  input(rep_ptr);

  fprintf(rep_ptr,"Results in a file\n");
  fprintf(end_ptr,"# Last generation population (Feasible and non-
dominated)\n");
  fprintf(end_ptr,"# Fitness_vector (first %d)  Constraint_violation
(next %d)  Overall_penalty\n",nfunc,ncons);
  fprintf(g_var,"#Feasible Variable_vectors for non-dominated
solutions at last generation\n");
  fprintf(g_var,"# Real (first %d)  Binary (next %d)\n",nvar,nchrom);
  fprintf(lastit,"# Feasible and Non-dominated Objective Vector\n");

  /*Initialize the random no generator*/
  warmup_random(seed);

   /*Binary Initializaton*/
  if (nchrom > 0)
    init(old_pop_ptr);
  if (nvar > 0)
    realinit(old_pop_ptr);

  old_pop_ptr = &(oldpop);

  // decode binary strings
  decode(old_pop_ptr);

  old_pop_ptr = &(oldpop);
  new_pop_ptr = &(newpop);

  for(j = 0;j < popsize;j++)
    {
      /*Initializing the Rank array having different individuals
      at a particular  rank to zero*/
       old_pop_ptr->rankno[j] = 0;
       new_pop_ptr->rankno[j] = 0;
    }

  old_pop_ptr = &(oldpop);

  func(old_pop_ptr);
  /*Function Calculaiton*/

  fprintf(rep_ptr,"----------------------------------------------------
--\n");
  fprintf(rep_ptr,"Statistics at Generation 0 ->\n");
  fprintf(rep_ptr,"----------------------------------------------------
\n");


/*******************************************************************
/
  /*----------------------GENERATION STARTS HERE---------------------
-*/
  for (i = 0;i < gener;i++)
    {
      printf("Generation = %d\n",i+1);
      old_pop_ptr = &(oldpop);
      mate_pop_ptr = &(matepop);
      fprintf(rep_ptr,"Population at generation no. -->%d\n",i+1);
      fprintf(gen_ptr,"#Generation No. -->%d\n",i+1);
      fprintf(gen_ptr,"#Variable_vector  Fitness_vector
Constraint_violation Overall_penalty\n");

      /*--------SELECT----------------*/
```

```
nselect(old_pop_ptr ,mate_pop_ptr );

new_pop_ptr = &(newpop);
mate_pop_ptr = &(matepop);

/*CROSSOVER----------------------------*/
if (nchrom > 0)
{

  if(optype == 1)
    {
      crossover(new_pop_ptr ,mate_pop_ptr );
      /*Binary Cross-over*/
    }

  if(optype == 2)
    {
      unicross(new_pop_ptr ,mate_pop_ptr );
      /*Binary Uniform Cross-over*/
    }
}
if (nvar > 0)
realcross(new_pop_ptr ,mate_pop_ptr );
/*Real Cross-over*/


/*------MUTATION-------------------*/
new_pop_ptr = &(newpop);

if (nchrom > 0)
mutate(new_pop_ptr );
/*Binary Mutation */

if (nvar > 0)
real_mutate(new_pop_ptr );
/*Real Mutation*/

new_pop_ptr = &(newpop);

/*-------DECODING----------*/
if(nchrom > 0)
decode(new_pop_ptr );
/*Decoding for binary strings*/

/*----------FUNCTION EVALUATION-----------*/
new_pop_ptr = &(newpop);
func(new_pop_ptr );

/*-------------------SELECTION KEEPING FRONTS ALIVE----------*/
old_pop_ptr = &(oldpop);
new_pop_ptr = &(newpop);
mate_pop_ptr = &(matepop);

/*Elitism And Sharing Implemented*/
keepalive(old_pop_ptr ,new_pop_ptr ,mate_pop_ptr,i+1);

mate_pop_ptr = &(matepop);
if(nchrom > 0)
decode(mate_pop_ptr );

mate_pop_ptr = &(matepop);
```

```
      /*------------------REPORT PRINTING--------------------------*/
      report(i ,old_pop_ptr ,mate_pop_ptr ,rep_ptr ,gen_ptr, lastit
);


/*===================================================================*
/

      /*-----------------Rank Ratio Calculation---------------------*/
      new_pop_ptr = &(matepop);
      old_pop_ptr = &(oldpop);

      /*Finding the greater maxrank among the two populations*/

      if(old_pop_ptr->maxrank > new_pop_ptr->maxrank)
      maxrank1 = old_pop_ptr->maxrank;
      else
      maxrank1 = new_pop_ptr->maxrank;

      fprintf(rep2_ptr,"--------RANK AT GENERATION %d-------\n",i+1);
      fprintf(rep2_ptr,"Rank old ranks   new ranks     rankratio\n");

      for(j = 0;j < maxrank1 ; j++)
      {
        /*Sum of the no of individuals at any rank in old population
          and the new populaion*/

        tot = (old_pop_ptr->rankno[j])+ (new_pop_ptr->rankno[j]);

        /*Finding the rank ratio for new population at this rank*/

        new_pop_ptr->rankrat[j] = (new_pop_ptr->rankno[j])/tot;

        /*Printing this rank ratio to a file called ranks.dat*/

        fprintf(rep2_ptr," %d\t  %d\t\t %d\t %f\n",j+1,old_pop_ptr-
>rankno[j],new_pop_ptr->rankno[j],new_pop_ptr->rankrat[j]);

      }

      fprintf(rep2_ptr,"-----------------Rank Ratio-------------\n");

/*===================================================================*
/

      /*=======Copying the new population to old population======*/

      old_pop_ptr = &(oldpop);
      new_pop_ptr = &(matepop);

      for(j = 0;j < popsize;j++)
      {
        old_pop_ptr->ind_ptr = &(old_pop_ptr->ind[j]);
        new_pop_ptr->ind_ptr = &(new_pop_ptr->ind[j]);
        if(nchrom > 0)
          {
            /*For Binary GA copying of the chromosome*/

            for(l = 0;l < chrom;l++)
            old_pop_ptr->ind_ptr->genes[l]=new_pop_ptr->ind_ptr-
>genes[l];
```

```
            for(l = 0;l < nchrom;l++)
            old_pop_ptr->ind_ptr->xbin[l] = new_pop_ptr->ind_ptr-
>xbin[l];
          }
        if(nvar > 0)
          {
            /*For Real Coded GA copying of the chromosomes*/
            for(l = 0;l < nvar;l++)
            old_pop_ptr->ind_ptr->xreal[l] = new_pop_ptr->ind_ptr-
>xreal[l];
          }

        /*Copying the fitness vector */
        for(l = 0 ; l < nfunc ;l++)
          old_pop_ptr->ind_ptr->fitness[l] = new_pop_ptr->ind_ptr-
>fitness[l];

        /*Copying the dummy fitness*/
        old_pop_ptr->ind_ptr->cub_len = new_pop_ptr->ind_ptr-
>cub_len;

        /*Copying the rank of the individuals*/
        old_pop_ptr->ind_ptr->rank = new_pop_ptr->ind_ptr->rank;

        /*Copying the error and constraints of the individual*/

        old_pop_ptr->ind_ptr->error = new_pop_ptr->ind_ptr->error;
        for(l = 0;l < ncons;l++)
          {
            old_pop_ptr->ind_ptr->constr[l] = new_pop_ptr->ind_ptr-
>constr[l];
          }

        /*Copying the flag of the individuals*/
        old_pop_ptr->ind_ptr->flag = new_pop_ptr->ind_ptr->flag;
      }   // end of j

      maxrank1 = new_pop_ptr->maxrank ;

      /*Copying the array having the record of the individual
      at different ranks */
      for(l = 0;l < popsize;l++)
      {
        old_pop_ptr->rankno[l] = new_pop_ptr->rankno[l];
      }

      /*Copying the maxrank */
      old_pop_ptr->maxrank = new_pop_ptr->maxrank;

      /*Printing the fitness record for last generation in a file
last*/
      if(i == gener-1)
            {  // for the last generation
        old_pop_ptr = &(matepop);
        for(f = 0;f < popsize ; f++) // for printing
          {
            old_pop_ptr->ind_ptr = &(old_pop_ptr->ind[f]);
            if ((old_pop_ptr->ind_ptr->error <= 0.0) && (old_pop_ptr-
>ind_ptr->rank == 1))  // for all feasible solutions and non-
dominated solutions
            {
              for(l = 0;l < nfunc;l++)
                fprintf(end_ptr,"%f\t",old_pop_ptr->ind_ptr-
>fitness[l]);
              for(l = 0;l < ncons;l++)
                {
                  fprintf(end_ptr,"%f\t",old_pop_ptr->ind_ptr-
>constr[l]);
                }
              if (ncons > 0)
                fprintf(end_ptr,"%f\t",old_pop_ptr->ind_ptr->error);
              fprintf(end_ptr,"\n");

              if (nvar > 0)
                {
                  for(l = 0;l < nvar ;l++)
                  {
                    fprintf(g_var,"%f\t",old_pop_ptr->ind_ptr-
>xreal[l]);
                  }
                  fprintf(g_var,"  ");
                }

              if(nchrom > 0)
                {
                  for(l = 0;l < nchrom;l++)
                  {
                    fprintf(g_var,"%f\t",old_pop_ptr->ind_ptr-
>xbin[l]);
                  }
                }
              fprintf(g_var,"\n");
            }  // feasibility check
          } // end of f (printing)

      } // for the last generation
    }  // end of i

  /*                   Generation Loop Ends
*/

/*******************************************************************/

  fprintf(rep_ptr,"NO. OF CROSSOVER = %d\n",ncross);
  fprintf(rep_ptr,"NO. OF MUTATION = %d\n",nmut);
  fprintf(rep_ptr,"---------------------------------------------------
----------\n");
  fprintf(rep_ptr,"---------------------------------Thanks-----------
----------\n");
  fprintf(rep_ptr,"---------------------------------------------------
-----------\n");
  printf("NOW YOU CAN LOOK IN THE FILE OUTPUT2.DAT\n");

  /*Closing the files*/
  fclose(rep_ptr);
  fclose(gen_ptr);
  fclose(rep2_ptr);
  fclose(end_ptr);
  fclose(g_var);
  fclose(lastit);
}
```

Listing B.1: Non-dominated Sorting Genetic Algorithm-II (NSGA-II) listing

```
/*This is the program used to evaluate the value of the function &
errors
*****************************************************************/
#include <iostream.h>
#include <stdio.h>
#include <fstream.h>
#include <cstdlib>
#include <ctime>

/*#define ofstream STD_OFSTREAM*/

void func(population *pop_ptr);
void func(population *pop_ptr)

{
/*File ptr to the file to store the value of the g for last iteration
    g is the parameter required for a particular problem
    Every problem is not required*/

  float *realx_ptr, /*Pointer to the array of x values*/
    *binx_ptr,      /* Pointer to the binary variables */
    *fitn_ptr,      /*Pointer to the array of fitness function*/
    x[2*maxvar],     /* problem variables */
    f[maxfun],     /*array of fitness values*/
    *err_ptr,      /*Pointer to the error */
    cstr[maxcons];

  float *ptr;
  FILE
    *param_ptr,    // parameter file
      *res1_ptr,  // result1 file
      *res2_ptr,  // result2 file
      *res3_ptr;
  /*File Pointers*/

  int i,j,k;
  float error, cc;
  float sum = 0;
  float res1, res2, res3;
  //ofstream paramfile;

  pop_ptr->ind_ptr= &(pop_ptr->ind[0]);

  /*Initializing the max rank to zero*/
  pop_ptr->maxrank = 0;
  for(i = 0;i < popsize;i++)
    {
      pop_ptr->ind_ptr = &(pop_ptr->ind[i]);
      realx_ptr = &(pop_ptr->ind_ptr->xreal[0]);
      binx_ptr = &(pop_ptr->ind_ptr->xbin[0]);
      //printf ("variables : %d \n", realx_ptr);

      for(j = 0; j < nvar; j++)
      { // Real-coded variables
        x[j] = *realx_ptr++;
        //sum = sum + x[j];
        //printf ("variables : %f\n" , x[j]);

      }

      for(j = 0; j < nchrom; j++)
      { // Binary-codced variables
        x[nvar+j] = *binx_ptr++;
      }

      fitn_ptr = &(pop_ptr->ind_ptr->fitness[0]);
      err_ptr = &(pop_ptr->ind_ptr->error);



      /*   DO NOT CHANGE ANYTHING ABOVE   */
      /*----------------------CODE YOUR OBJECTIVE FUNCTIONS HERE---*/
      /*All functions must be of minimization type, negate
maximization functions */
      /*==Start Coding Your Function From This Point=======*/
      // First fitness function
        param_ptr = fopen("param.txt", "w");   // parameter file

        fprintf(param_ptr,"%f\n%f\n%f\n%f\n",x[0],x[1],x[2],x[3]);
        fclose(param_ptr);
        system("perl L:\\MyFolder\\MyPhd\\MOO_NSGA\\ota_pareto.pl");

        res1_ptr = fopen("result_gm.txt", "r"); //result func 1 file
        res2_ptr = fopen("result_ro.txt", "r"); // result func 2 file
        res3_ptr = fopen("result_pm.txt", "r"); // result func 3 file
        fscanf(res1_ptr, "%f", &res1);
        fscanf(res2_ptr, "%f", &res2);
        fscanf(res3_ptr, "%f", &res3);


        f[0] = res1;
        f[1] = res2;
        f[2] = res3;

      /*=========End Your Coding Upto This Point===============*/


/*******************************************************************/
/*              Put The Constraints Here                          */

/*******************************************************************/
      // g(x) >= 0 type (normalize g(x) as in the cstr[1] below)
      /*===========Start Coding Here=============*/

      cstr[0] = x[0]*x[0]+x[1]*x[1]-1.0-
0.1*cos(16.0*atan(x[0]/x[1]));
      cstr[1] = (-square(x[0]-0.5) - square(x[1]-0.5) + 0.5)/0.5;

      /*===========Constraints Are Coded Upto Here=============*/
      /*   DO NOT CHANGE ANYTHING BELOW  */



      for(k = 0 ; k < nfunc ;k++)
      {
        *fitn_ptr++  = f[k];
      }

      for (k = 0;k < ncons; k++)
      {
        pop_ptr->ind_ptr->constr[k] = cstr[k];
      }
      error = 0.0;
      for (k = 0;k < ncons;k++)
      {
        cc = cstr[k];
        if(cc < 0.0)
          error = error - cc;
      }
      *err_ptr = error;
    }

  /*---------------------------* RANKING *------------------------*/

  if(ncons == 0)
    ranking(pop_ptr);
  else
    rankcon(pop_ptr);

  return;
}
```

Listing B.2: NSGA-II function evaluation listing

Appendix C: Verilog-A Model Listings

```
`include "constants.vams"
`include "disciplines.vams"

module ota(out, inp, inm);

      inout inp, inm;
      output out;

      electrical inp, inm, out;

      parameter real gm = 136u;
      parameter real gm3 = 50u;
      parameter real ro = 106.2k;
      parameter real ce = 126f;
      parameter real cgd1 = 15f;
      parameter real cout = 150f;
      parameter real cgd2 = 15f;

      real vin;
      electrical vm;

      analog begin


            vin = V(inp,inm);
            I(vm) <+ -gm*(vin/2);
            I(vm) <+ V(vm)/(1/gm3);
            I(vm) <+ ce*ddt(V(vm));
            I(vm) <+ cgd1*ddt(vin/2);

            I(out) <+ -gm3*V(vm);
            I(out) <+ -gm*vin/2;
            I(out) <+ cout*ddt(V(out));
            I(out) <+ cgd2*ddt(vin/2);
            I(out) <+ V(out)/ro;



      end

endmodule
```

Listing C.1: Verilog-A Single Stage OTA listing

```
`include "constants.vams"
`include "disciplines.vams"

module vco(in, out);

  inout in, out;
  electrical in, out;


  parameter real fmin = 300e6 from (100e6:80e7); //hertz
  parameter real fmax = 500e6 from (200e6:40e8); //hertz
  parameter real Ivco = 13.2e-3 from (1e-3:30e-3);
  parameter real lfwh = 0.0 from [0:1.0);

  real ko;
  real lffl;
  real vmax;
  real vmin;

  analog begin

  vmax=1.2;
  vmin=0.2;

  ko = (fmax-fmin)/(vmax-vmin);

  lffl = $table_model(ko,Ivco, "vco_data.tbl", "3L,3L");


V(out) <+ laplace_nd(V (in), {fmax-fmin/1},{0,1})
        + flicker_noise(lffl, 3, "VCO_flicker")
        + flicker_noise(lfwh, 2, "VCO_white");

end
endmodule
```

Listing C.1: Verilog-A VCO listing

```
// VCO variation module for minimum

`include "constants.vams"
`include "disciplines.vams"

module vco_min(in, out);

  inout in, out;
  electrical in, out;


  parameter real fmin = 300e6 from (100e6:80e7); //hertz
  parameter real fmax = 500e6 from (200e6:40e8); //hertz
  parameter real Ivco = 13.2e-3 from (1e-3:30e-3);
  parameter real lfwh = 0.0 from [0:1.0);

  real ko;
  real ko_min;
  real Ivco_min;
  real lffl;
  real vmax;
  real vmin;
  real min_fmin, min_fmax;
  real lffl_min;

  integer file_ptr1,file_ptr2,file_ptr3,file_ptr4;

  analog begin

  vmax=1.2;
  vmin=0.2;


   @(initial_step) begin
  file_ptr1 = $fopen("ivcomin.txt");
  file_ptr2 = $fopen("minfmin.txt");
  file_ptr3 = $fopen("minfmax.txt");
  file_ptr4 = $fopen("komin.txt");
  end

  //minimum variation for Ivco
  Ivco_min = $table_model(Ivco, "Ivcomin_data.tbl", "3L");
  $fwrite(file_ptr1, "%e", Ivco_min);

  // minimum variation for fmin and fmax
  min_fmin = $table_model(fmin, "fmin_mindata.tbl", "3L");
  $fwrite(file_ptr2, "%e", min_fmin);
  min_fmax = $table_model(fmax, "fmax_mindata.tbl", "3L");
  $fwrite(file_ptr3, "%e", min_fmax);

  // minimum variation for ko
  ko_min = (min_fmax-min_fmin)/(vmax-vmin);
  $fwrite(file_ptr4, "%e", ko_min);

  ko = (fmax-fmin)/(vmax-vmin);

  lffl = $table_model(ko,Ivco, "vco_data.tbl", "3L,3L");

  // minimum variation for lffl noise
  lffl_min = $table_model(lffl, "lfflmin_data.tbl", "3L");


  V(out) <+ laplace_nd(V (in), {(fmax-fmin)/1},{0,1})
        + flicker_noise(lffl_min, 3, „VCO_flicker‟)
        + flicker_noise(lfwh, 2, ‟VCO_white”);

  $fclose(file_ptr1);
  $fclose(file_ptr2);
  $fclose(file_ptr3);
  $fclose(file_ptr4);

end
endmodule
```

Listing C.2: Verilog-A VCO minimum variation listing

```
// VCO variation module for maximum

`include "constants.vams"
```

```
`include "disciplines.vams"

module vco_max(in, out);

  inout in, out;
  electrical in, out;


  parameter real fmin = 300e6 from (100e6:80e7); //hertz
  parameter real fmax = 500e6 from (200e6:40e8); //hertz
  parameter real Ivco = 13.2e-3 from (1e-3:30e-3);
  parameter real lfwh = 0.0 from [0:1.0);

  real ko;
  real ko_max;
  real Ivco_max;
  real lffl;
  real vmax;
  real vmin;
  real max_fmin, max_fmax;
  real lffl_max;


  integer file_ptr1,file_ptr2,file_ptr3,file_ptr4;


  analog begin

  vmax=1.2;
  vmin=0.2;


  @(initial_step) begin
  file_ptr1 = $fopen("ivco.txt");
  file_ptr2 = $fopen("maxfmin.txt");
  file_ptr3 = $fopen("maxfmax.txt");
  file_ptr4 = $fopen("komax.txt");
  end
  // maximum variation for Ivco
  Ivco_max = $table_model(Ivco, "Ivcomax_data.tbl", "3L");
  $fwrite(file_ptr1, "%e", Ivco_max);


  // maximum variation for fmin and fmax
  max_fmin = $table_model(fmin, "fmin_maxdata.tbl", "3L");
  $fwrite(file_ptr2, "%e", max_fmin);
  max_fmax = $table_model(fmax, "fmax_maxdata.tbl", "3L");
  $fwrite(file_ptr3, "%e", max_fmax);

  // maximum variation for ko
  ko_max = (max_fmax-max_fmin)/(vmax-vmin);
  $fwrite(file_ptr4, "%e", ko_max);

  ko = (fmax-fmin)/(vmax-vmin);

  lffl = $table_model(ko,Ivco, "vco_data.tbl", "3L,3L");

  // maximum variation for lffl noise
  lffl_max = $table_model(lffl, "lfflmax_data.tbl", "3L");
  V(out) <+ laplace_nd(V (in), {(fmax-fmin)/1},{0,1})
        + flicker_noise(lffl_max, 3, "VCO_flicker")
        + flicker_noise(lfwh, 2, "VCO_white");

  $fclose(file_ptr1);
  $fclose(file_ptr2);
  $fclose(file_ptr3);
  $fclose(file_ptr4);

end
endmodule
```

Listing C.3: Verilog-A VCO maximum variation listing

```
`include "constants.vams"
`include "disciplines.vams"


module pfd(in1, in2, out);
  inout in1, in2, out;
  electrical in1, in2, out;

  parameter real Icp = 12e-6 from(0:1.0);
  parameter real lfpfl = 0.0 from [0:1.0);

  real kd;
  real lfpwh;

analog begin

  kd = Icp/(2*3.14);

  //lookup table for pfd_cp noise
  lfpwh = $table_model(Icp, "pfd_data.tbl","1E");

  //$display("lfpwh_value =  %e",lfpwh);

  V(out) <+ kd*(V(in1) - V(in2))
        + flicker_noise(lfpfl, 1, "pfd_flicker")
        + white_noise(lfpwh, "pfd_white");

end
endmodule
```

Listing C.4: Verilog-A PFD/CP listing

```
`include "constants.vams"
`include "disciplines.vams"


module pfd_min(in1, in2, out);
  inout in1, in2, out;
  electrical in1, in2, out;

  parameter real Icp = 12e-6 from(0:1.0);
  parameter real lfpfl = 0.0 from [0:1.0);
  real kd;
  real lfpwh;
  real lfpwhmin;

  integer file_ptr1;

analog begin

  @(initial_step) begin
  file_ptr1 = $fopen("icp.txt");
  end

  $fwrite(file_ptr1, "%e", Icp);

  kd = Icp/(2*3.14);

  //lookup table for pfd_cp noise
  lfpwh = $table_model(Icp, "pfd_data.tbl","1E");

  //lookup table for pfd variation
  lfpwhmin = $table_model(lfpwh, "pfdmin_data.tbl", "3L");

  V(out) <+ kd*(V(in1) - V(in2))
        + flicker_noise(lfpfl, 1, "pfd_flicker")
        + white_noise(lfpwhmin, "pfd_white");

  $fclose(file_ptr1);

end
endmodule
```

Listing C.5: Verilog-A PFD/CP minimum variation listing

```
`include "constants.vams"
`include "disciplines.vams"


module pfd_max(in1, in2, out);
  inout in1, in2, out;
  electrical in1, in2, out;

  parameter real Icp = 12e-6 from(0:1.0);
  parameter real lfpfl = 0.0 from [0:1.0);

  real kd;
  real lfpwh;
  real lfpwhmax;

  integer file_ptr1;

analog begin

  @(initial_step) begin
  file_ptr1 = $fopen("icp.txt");
  end

  $fwrite(file_ptr1, "%e", Icp);

  kd = Icp/(2*3.14);
  //lookup table for pfd_cp noise
  lfpwh = $table_model(Icp, "pfd_data.tbl","1E");

  //lookup table for pfd variation for maximum
  lfpwhmax = $table_model(lfpwh, "pfdmax_data.tbl", "3L");

  V(out) <+ kd*(V(in1) - V(in2))
        + flicker_noise(lfpfl, 1, "pfd_flicker")
        + white_noise(lfpwhmax, "pfd_white");

   $fclose(file_ptr1);

end
endmodule
```

Listing C.6: Verilog-A PFD/CP maximum variation listing

```
`include "constants.vams"
`include "disciplines.vams"

module PLL_top(ref_in, pll_out);

inout ref_in, pll_out;
electrical ref_in, pll_out;

  parameter real Icp = 10e-6 from(0:1.0);
  parameter real lfpfl = 0.0 from [0:1.0);

  parameter real C_1 = 1.0e-12 from (0:1.0e-3);
  parameter real R_2 = 10.0e3 from (0:1M);
  parameter real C_2 = 3.0e-12 from (0:1.0e-3);

  parameter real fmin = 300e6 from (100e6:80e7); //hertz
  parameter real fmax = 500e6 from (200e6:40e8); //hertz
  parameter real Ivco = 13.2e-3 from (1e-3:30e-3);
  parameter real lfwh = 0.0 from [0:1.0);

  parameter real ratio = 1 from (0:inf);


pfd # (.Icp(Icp), .lfpfl(lfpfl))
pfd1(ref_in, divout, filin);
loopfilter # (.C_1(C_1), .R_2(R_2), .C_2(C_2))
loopfilter1(filin, vcoin);
vco # (.fmin(fmin), .fmax(fmax), .Ivco(Ivco), .lfwh(lfwh))
vco1(vcoin, pll_out);
div # (.ratio(ratio))
divider1(pll_out, divout);

endmodule
```

Listing C.7: Verilog-A PLL top level listing