MASTER

# An analog synthesis tool

van Raaij, R.R.M.

*Award date:*
1988

Eindhoven Technical University
Department of Electrical Engineering
Digital Systems Group

# An Analog Synthesis Tool

Robin van Raaij, 179484

Created: 24 august 1988

# Contents

# Chapter 1

# Summary & Introduction

During my study to become an electrotechnical engineer, I developed a special interest in the so called system-technology. System-technology is directed towards the design process as a whole and its object is to control every step taken in the design process, from idea to realisation.

Somewhere in june 1987 I approached professor Dr. Ir. C.J. Koomen for a graduation project in my area of interest, which resulted in a project on analog synthesis. Analog synthesis is a term used to describe meant the process of generating an analog circuit with the aid of available circuit knowledge. Some analog designs, (e.g. opamp's, D/A and A/D convertors, analog filters), are built from a limited number of transistor structures of which a profound knowledge is available. An expert designer knows how and where to use these structures best. When the basic construction of a circuit is determined, the only task left is the sizing of the components. With the available knowledge, the whole process should be suited to automize.
It was my task to design and build a system that automatically, with the use of knowledge based techniques, could generate a class of operational amplifiers (opamp's) which could stand the comparison with an opamp designed by an expert designer.
The task of designing an opamp usually takes a period of several weeks and in case of a nonexperienced designer, several months. With the system this should be reduced to a few hours, of course without failures.
Special attention should be given on how to describe the knowledge in a knowledge-base. Furthermore, if possible, the existing framework of the knowledge based verification program, VERA [C-29] should be used.

In doing this project I started with a background study on knowledge based systems, electronic circuits and synthesis systems. A background on analog synthesis tools is given in chapter 2.
A framework for a system was proposed, based on ideas gathered out of literature, discussions with my attendants and my own ideas. This framework was discussed and accepted. A description of the framework is found in chapter 3.
Before starting to implement the system, I first investigated what kind and to what extent a designer was able to generate information concerning a design. Then it turned out that not enough information was available from the designers. Especially reliable circuit behaviour describing formulas (equations), with which the circuit components could be sized, could not be given. At that moment I had to decide on what to do: build a framework which was in fact worthless, because no reliable designs could be made with it, or start with an analysis tool to generate the necessary equations.
I chose to start with the analysis tool which resulted in a routine that, based on the principle of nodal network analysis, could derive the equations needed. This analysis tool, called Formula, is described in chapter 4. With Formula it is possible to derive circuit behaviour describing equations.
The next problem that occurred was how to solve a set of parameter equations which had a set of conflicting constraints. The solution implemented, described in chapter 5, is based on a combination of the so called Simplex method of Nelder and Mead, not to be mistaken with the Simplex method for linear optimization and Factor variation. Factor variation is a simple routine that varies the variable proportion in order to find the best proportion. With these combined methods it was possible to find a reasonable solution thereby synthesizing a circuit. However, it has to be remarked that more tests and experiments, on different circuits, have to be performed before this conclusion can be drawn definitely.

When this conclusion is confirmed, a tool can be built using these routines, with which at least machine done redesign of small circuits has become possible. The final conclusions on the project are drawn in chapter 6.

Officially the project was called "Knowledge based synthesis of operational amplifiers" and was conducted at the Philips Research Laboratories (the Natlab) at Eindhoven, the Netherlands.
In doing this project I collaborated with three researchers. The two expert

designers, Ir. Ivo Rutten (group Wouda) and Rob van de Grift (microtel), were willing to let me use their knowledge on the functioning of analog circuits. Both are expert designers in the area of analog circuit design and momentarily working at the Natlab. As my supervisor, Drs. Ton Kostelijk (group Nederlof) was responsible for the work I conducted. Much help was given by him, especially on the algorithms.
All three were enthusiastic discussion partners and good and helpful critics of my ideas. Here I would like to thank them for their enthusiasm and comradeship during the time I worked with them.
Also I would like to thank Ir. Guido Schrooten (group Nederlof) for his help during the first discussions, (he departed after a few months to work at Briarcliff Manor), and Philips for giving me the opportunity to work in such a high standing environment.
Last but not least I thank professor Koomen who, inspite of being very busy and always running to another appointment, in one way or another managed to find the time to discuss my work with me.

Robin van Raaij, Eindhoven, augustus 1988.

# Chapter 2

# Analog synthesis tools: an introduction

## 2.1 Why analog synthesis tools?

The why of an analog synthesis design tool is an easy tale to tell.
Why bother with analog circuitry at all one could ask, since even the more classical analog applications, like analog filtering, can be replaced by digital techniques. Furthermore the difference between analog and digital is diminishing because it is recognized that many analog functions can be realized with digital circuitry. Sometimes the digital solution is even better than the analog one [A-11].
There is a limit to this replacement process of analog by digital circuitry. There are many places where analog circuitry is needed to interface between the digital processing systems and the external environment where input or output is continuous. A continuous variable voltage or current level cannot be matched by digital circuitry. For instance in robotics and telecommunication there is a large demand for analog interfaces.
Processing technology is developing rapidly thereby decreasing the minimum feature size of devices. This allows complex systems to be integrated on one chip. Due to the system complexity, it is necessary to integrate analog and digital functions on one chip. For this reason implementation of analog functions in for example MOS-technology has become increasingly important. Much effort has been put into implementing functions such as high speed DAC's and sampled data analog filters.
In many cases very high performance analog functions are not that much required and a great number of moderate performance analog functions is

sufficient.

Until recently almost every analog function had to be designed completely by hand. It made no difference if it was a new design or a redesign according to slightly different specifications. A designer starts with a circuit which he expects to meet the specifications. This first guess is based on the experts knowledge and experience. Then the circuit is simulated and after interpreting the results the designer decides what changes are necessary whereupon he simulates again. This is a very time consuming process.

The reasons for building an analog circuit synthesis tool are:

- The manpower limitation, there are just a few analog experts that are able to understand and solve the problems of high perfomance analog functions.
  Further development of the process technology contributes to this problem by causing the increase of the transistor model complexity.
- Most designs done are relatively simple, for which a high expertise level is not needed.

An anlog synthesis tool can take the burden from the expert in trivial "done before" designs or redesigns so that the expert can concentrate on problems where the expertise is best used.
For often used analog circuitry e.g. opamps, D/A-A/D converters, a number of circuit structures can be used which are well known by the expert. A profound knowledge of these structures is available. Also the expert has knowledge about the application domain. Then, when the basic circuit is determined, only the sizing of the devices is left to be done.
With the available knowledge this process is in principle suited for automation.

## 2.2 Background information

It is only recently that design automation has begun to enter the analog circuit domain and so the state of analog synthesis tools is quite immature. Up to now, several implemented frameworks have been presented in literature which more or less seem to work. In contrast there exist several good working digital circuit synthesis tools.
First I will give some background information by reviewing the differences

between digital and analog circuitry and explain different approaches towards analog synthesis tools.

If we consider the task of designing a functional block, to be implemented on a VLSI circuit, then the overall highlevel synthesis task for either analog or digital circuits is to interconnect a set of thoroughly designed components in order to meet the functional specification of a block. The differences between analog and digital design tasks can be classified into four categories: size, hierarchy, process constraints and performance constraints [A-9].

Size:
Analog circuits tend to have fewer transistors than digital circuits. Nevertheless, much more time is needed to design an analog circuit because of the time needed to design the individual transistors. The difference is brought about by the assumed functionality and reusability of an individual transistor.
A transistor in a digital circuit functions purely as a switch and the design constraints for such a transistor are related to switch and delay times. Furthermore almost all transistors in a digital circuit are identical. Only those transistors that have to perform a special task, like driving large loads, have to be specially designed.
In analog circuits the circuit behaviour is strongly influenced by the interconnection of the individual components which the circuit is built of. It is also possible that an individual transistor performs more than one function. On top of that, the function of a transistor, in relation with the circuit behaviour, only becomes clear during the design process or even after the design is completed. Therefore the electrical characteristics of every device have to be designed very carefully.

Hierarchy:
When we observe the design process of an analog and a digital circuit we notice for both a hierarchical way of doing the design.
For digital circuits there is a general agreement about how and which steps to take to get from specification to realisation:

- global function description
- translation to function blocks descriptions
- translation to gate networks
- translation to switch networks

- circuit designs
- mask design

For analog circuits, there exists not such an agreement. However, there is a hierarchy although the abstraction levels are not quite clear. For instance for digital circuits, at the highest abstraction level, we simply ignore the electrical characteristics of signals and take them into account on a lower level. With analog designs even at the highest level we have to think about low-level electrical characteristics. A different thing is, that a complex circuit like an opamp in one situation might consist of one transistor while in another situation might consist of 20 resulting in a completely different hierarchical approximation of the design.
These extremities appear less frequently in digital designs.

Process constraints:
At the toplevel of the digital circuit design hierarchy the process constraints are taken into account in a simplified form. The constraints are known and there are possibilities to overcome problems created by these constraints. With analog design the constraints are much more stringent. If for a certain functional block, in a chosen design style, components are required to match specific electrical characteristics which are not feasible in the process, then it is simply useless to continue the design and another design style has to be selected.
The design of the individual device characteristics is largely based on process parameters. Inaccurate consideration of these process parameters can ruin the performance of the whole circuit. This effect is much stronger during the device by device design for analog circuits.

Performance constraints:
Also in this case there is a great difference in analog and digital design.
The behaviour of a digital circuit is specified by a purely functional description. For analog circuits the behaviour is known implicitly e.g. an opamp amplifies, an A/D-convertor digitizes signals. The specifications of an analog circuit are usually given by a set of parameters such as gain, phase-margin, bandwidth, noise. These parameters constrain continuous quantities for example voltages or currents.

The differences between analog and digital circuit design make it obvious why analog design is much more difficult and time consuming than digital design.

## 2.3 Different synthesis approaches

Analog design is difficult. It is perhaps even more difficult to design a fully automated analog synthesis tool which generates reasonable designs that work not only according to the functional constraints but are also comparable with expert made designs. Several attempts have been made to create such a tool. In literature [A-1;A-16] one can distinguish three approaches:

- layout based approaches
- parameterized-structure approaches
- artificial-intelligence (AI), knowledge-based (KB) approaches

- layout based approaches:
Use of single-component arrays (= transistor or gate arrays) or circuit arrays (standard cell) in combination with place and route tools that use the electrical characteristics of the available devices to create a layout in silicon. The disadvantage of this approach is that, due to a limited set of device types that are predesigned and fixed in componentsize, the realizable performance variation is reduced. This means that the circuit design itself is constrained.

- parameterized-structure approaches:
This approach uses standard cells as building blocks. It differs from the layout-based approach in that the designer can parameterize some of the cells. In fact this works like a module generator because part of a circuit is fixed and the remainder is parameterized. This approach permits a wider range of performance specifications compared to the layout based approach but also limits the possibilities in circuit design.

- artificial-intelligence, knowledge based approaches:
Early work on application of artificial intelligence (AI) to analog circuit design was concentrated on two techniques. The first concentrated on circuit design starting from the component level. By using information about the terminal behaviour of a component and the effects and influences of the interconnecting pattern of components, a circuit design was made to yield a specific circuit specification. Due to the difficulty of relating component terminal behaviour toward the system function only a few simple circuits were designed.
This technique strongly depends on rules that were derived in understanding

the qualities of a component and the interconnecting pattern. Development of these rules is still in progress at the Technical University of Delft by E.H. Nordholt who is working in the group of professor J. Davidse. They try to find rules with which they, starting with a common emitter transistor, can build a circuit. They obtained some results but are still far from the point where they can design large circuits. The wilder the interconnecting pattern (e.g. feedforward or feedback loops) the more difficult to find a rule that is satisfactory.

The second technique was based on the use of circuit analysis programs like EL from Stallman and Sussman [A-1] or EQUAL from De Kleer [A-5]. Circuit analysis programs try, in reasoning about a circuit, to explain how the function of a circuit is related to its structure. EL uses heuristic inspection techniques to analyse the circuit by propagation of constraints. EQUAL is based on the intuitive qualitative reasoning that electrical engineers use when they analyse a circuit. The problem with tools that used this technique was that they were very good learning tools but unable to design a circuit according to the specifications. Infact they lacked the design rules which, among others, Nordholt is searching for.
It is said [A-9] that causal models and qualitative reasoning advance AI research more than analog synthesis research.

The expert-system technology seems more promising. Using this technology to automate the analog design process requires, that the human expert designers knowledge is captured and put in such a form that a machine can handle it. It is very difficult to capture this knowledge especially in this area were every expert has his own method to attack and solve a problem. The method also varies with the constraints a design has to meet. Although the reasoning patterns used by analog designers are not yet understood, it is observed that most designers start by subdividing the system into smaller blocks [C-36]. The more experience and knowledge an expert has, the better he is able to partition the system and to relate parts of the system with the constraints.
Several systems using the knowledge based approach more or less seem to work.

In designing a circuit, about 90% of the time is spent simulating the circuit and interpreting the simulation results. During the remaining 10% the designer is really creative in generating new ideas. This is recognized by some synthesis tool designers and so tools have been built that combine the

knowledge based approach with a simulator. The knowledge base contains rules to interpret the simulation results and to take appropriate steps. Tools like Delight-Spice [A-13] and OP1 [A-14] use this combination. In both cases the Spice simulator is used as a passive tool.

Next to these three approaches there exist programs that are written to redesign a specific cicuit according to different specifications. One such a program, SCOPA [C-37] is used in Philips to design series of Miller-opamps. This program contains an extensive description of the Miller-opamp circuit. It took about two years to get the necessary circuit information and to write the program. One of the conclusions which is drawn from the experience with this more specific approach is that it is a tedious job to do and on top of that, the program loses value because of process technology developments (see also section 1 of the next chapter).

## 2.4 Literature examples

I will briefly describe Blades, Oasys and Idac as examples of the knowledge based approach.

Blades stands for Bell Laboratories Analog Design Expert System [A-6,15].
The architecture is based on circuit partitioning and the use of subcircuits as standard building blocks. Circuits on component level are not designed.
The system consists of a general design expert manager, that coordinates the systems actions and subcircuit-experts, that contain detailed knowledge and basic circuits concerning a subcircuit type. Given a circuit specification the general design expert manager attempts to define a topology using the available building blocks. The subcircuits are then connected and tested. Necessary changes are made based on the analytical test results.
The number of stages and their internal architecture depends on the input specifications translation by global formulas.
The knowledge base Blades uses, consists mainly of a large collection of subcircuits, frequently used in analog designs.
The time needed to design an opamp is claimed to be approximately 20 minutes.

Oasys, a prototype OpAmp Synthesis System [A-9].
The architecture is based on a hierarchy by which first a fixed topology is selected and secondly by using a planning mechanism, the proper sized tran-

sistorized circuit is fit in. The preconditions for selecting a transistorized circuit are set by translation of the global specifications.
The knowledge base contains basic circuits, topologies and plans. Every topology has its own plan. In a plan the translations operations are described. For an opamp topology approximately 25 plan (= translation) steps and 10 plan modification steps are needed.
The 25 plan steps translate the overall constraints to transistor constraints while the modification steps modify the translation process in case a specification is not met. Oasys needs approximately 2 minutes to synthesize an opamp.

IDAC is a program for Interactive Design of Analog Circuits.[A-4]
Starting from technological data and building block specifications the program synthesizes a circuit in just a few seconds. The program consists of a knowledge base that contains circuits with relating formulas and a user-interface. By using the formulas the program attempts to meet the user specifications.

With respect to these three systems and the other systems mentioned in literature, only a few reports exist concerning analog synthesis designed circuits that are simulated, fabricated and tested. This means it is very difficult to value these systems. However in every respect it is a great advantage that some of the sytem proposals are implemented and tested since presenting a proposal for a global framework to perform analog synthesis is not the problem, implementing it in a way that it works is!

# Chapter 3

# A framework proposal

In this chapter I will motivate the reasons behind the system structure I proposed. First the system specifications are discussed, after which the framework is described.
Those who are acquinted with the literature on the subject of analog synthesis systems will notice some resemblance with existing (or proposed) system frameworks. That is no coincidence since I used part of the ideas, which I thought were very useful, in combination with my own observations and ideas. The result however, is again another point of view relating to the problem of analog synthesis.

## 3.1 System constraints

Although the task of designing and building a knowledge based synthesis system was aimed at opamps, the system should be structured in such a way that it could handle other circuits or circuit combinations also.
In order to determine the specifications of the synthesis system an idea for a larger framework was proposed by Rob van de Grift and Ivo Rutten. This idea was based on activities in other projects and projects planned. The idea, which I will not discuss extensively was called "an Architecture specific silicon compilation". The framework could be roughly divided into two major blocks: an electrical design (synthesis) block that synthesized an electrical circuit and an optimiser block that would finetune the circuit to the specifications (figure 3.1).

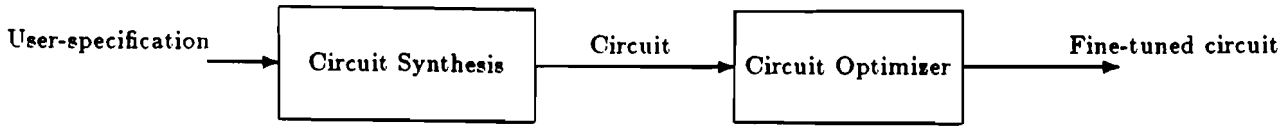



figure 3.1

During the discussions based on this framework, the following specifications were set for the synthesis system:

- The system contains general circuit knowledge, basic designs and basic formulas.
- The system inputs are: the user-specifications according to which a design should be done and process technology specifications.
- The system output is a circuit diagram with sized components that satisfies the user-specifications. The output was meant to be an almost certain good solution (a good first shot) with which the optimiser block could come towards a near perfect solution.

Perhaps one of the most important things to be kept in mind is the meaning of process technology on the value of a synthesis system. A separate technology input file is sufficient to make the system usable for different process technologies.
The speed with which process technology is developing is a more serious problem. If we describe a circuit we use certain models for the circuit components. A model can be viewed as a combination of capacitors, resistors and voltage controlled current sources, see figure 3.2. Depending on the process technology used and the influence of a component on circuit behaviour, some of the model components can be neglected. This is very practical if we want to derive formulas which describe the circuits functionality. When, due to development of the technology, the minimum feature size of devices in a technology shrink, the importance of the model components changes. As a result, model components that could be neglected before, can now become crucial for a correct description of the circuit. This means that the derived formulas, wherefore a simplified model was used, can be wrong. In

worst case all descriptions of the circuits become worthless and with it the synthesis system. More or less this is what happens with SCOPA [C-37], a computer program that generates series of Miller-opamps. So it is very important that the system should be indifferent or adjustable to technology changes.

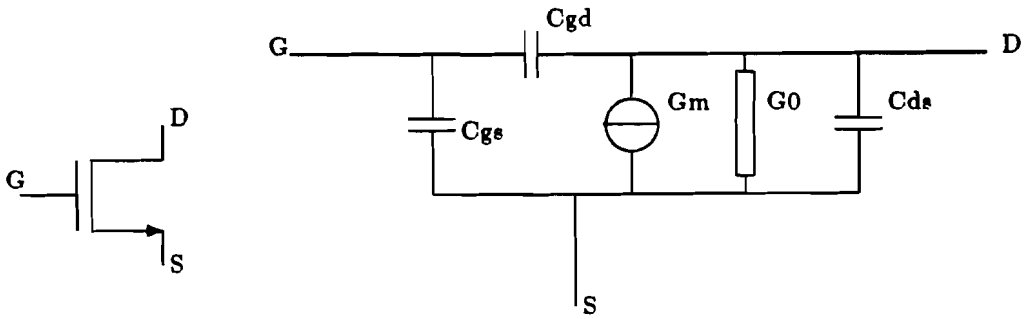



figure 3.2, MOS-transistor and transistor model

During discussions with the designers it turned out that, when designing a system that would contain circuits and circuit knowledge, it would not be such a bad idea to use the system for other things as well, namely as a kind of archive, not only for storing a complete design but also useful parts of it (e.g. current sources). The reason why is that the way designs are described and archived now is insufficient. First of all designers are not motivated to write an extensive report about their design because they are pressurized to start a new design a soon as possible. Furthermore, one simply forgets to write down important things because they seem trivial at the time. As a result, it is not uncommon that when a designer is asked to redesign his design according to slightly different specifications, it takes him quite a while before he has regained all the ins and outs of his design. It is even worse if the original designer has left the company and someone else has to do it. Also knowledge simply cannot be found because of the amount of knowledge available and the way it is saved.
In my opinion a synthesis system should only be used to archive knowledge that is vital for synthesizing circuits. Storing every designed circuit will create an enormous overhead, detoriating the systems performance. When a general archive function is needed, a dedicated sytem should be made for it.
When in the following the archive function is mentioned, only the storage

of knowledge vital to the system is meant.

Summarizing:

- the system should be able to perform the following functions:
  - design a circuit according to the specifications of a parameter set
  - redesign a circuit according to slightly different specifications
  - archive a circuit design, time needed to feed a new design into the system should be kept to a minimum
  - be able to generate separate structures on request, for instance current sources, out of archived designs to be reused by a designer
- the system input is:
  - process technology parameters
  - user specifications
- the systems output consists of a sized transistorised circuit
- the system must be indifferent or adjustable to technology changes

## 3.2 System proposal

In combining my own ideas and ideas obtained by studying literature a system configuration, shown in figure 3.3, was proposed with which synthesis was possible.
The framework is based on hierarchy in which a circuit can be viewed as a combination of separate building blocks. This stems not only from the observations as discussed in chapter 1. When reading articles or books on electronic circuit design, circuit and circuit behaviour are always explained by the use of so-called basic circuits. Although I realize that explaining a circuit is not a guarantee that it is also possible to construct a circuit that way, it is an argument for using such a hierachical approximation to synthesize circuits in a case where one uses known topologies. I chose for the use of known topologies because there exist no rules or methods with which, starting from scratch and according to some parameter constraints, a circuit can be constructed. On the contrary, for several circuit types e.g. opamps or switched-capacitor-filters, so much knowledge is available that enough fixed topologies can be found with which a broad spectrum of specifications can

be reached.
When using a fixed topology, the interconnection pattern of the building blocks that construct the circuit on top level is known. This pattern can be used to translate from top-level specifications towards specifications for the building blocks. According to these specifications a transistorized basic circuit can be selected.

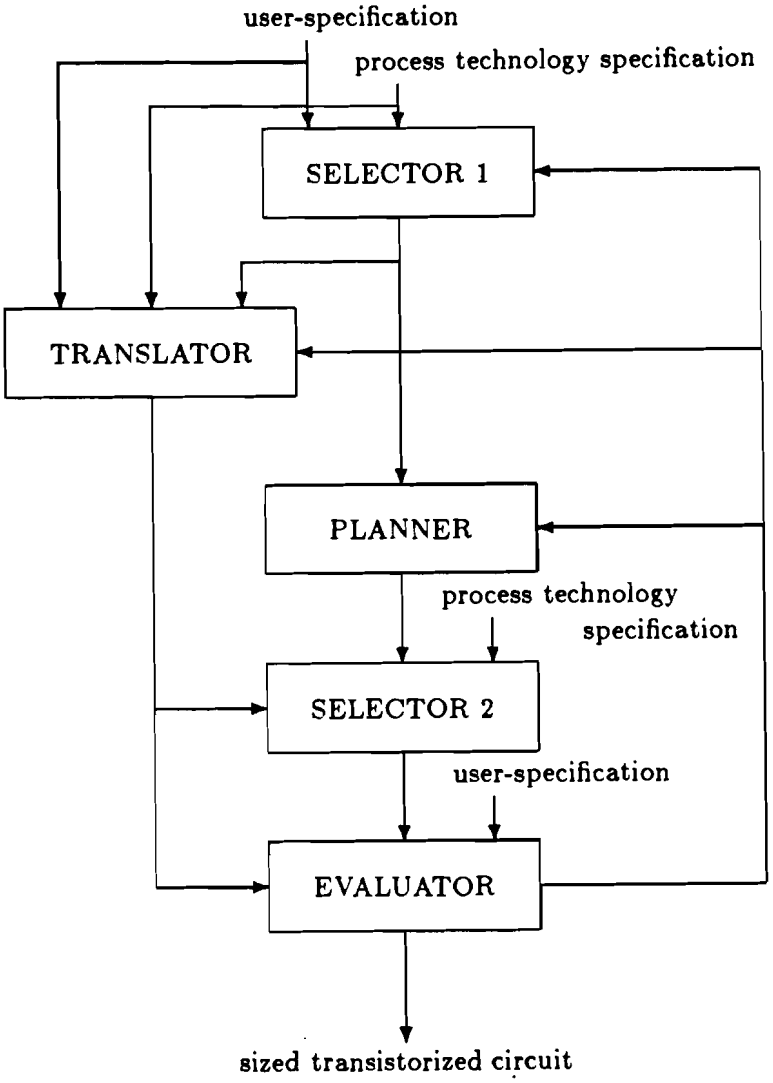



figure 3.3

In the following explanation the evaluator input on the separate blocks is ignored. The meaning of this input will be explained when the evaluator block is discussed. The knowledge in the knowledge base (KB) is partitioned into knowledge that is used to make a decision, called KB-rules (e.g. design rules) and knowledge out of which a selection can be made, called KB-facts (e.g. a set of basic circuits).

### 3.2.1 Selector-1

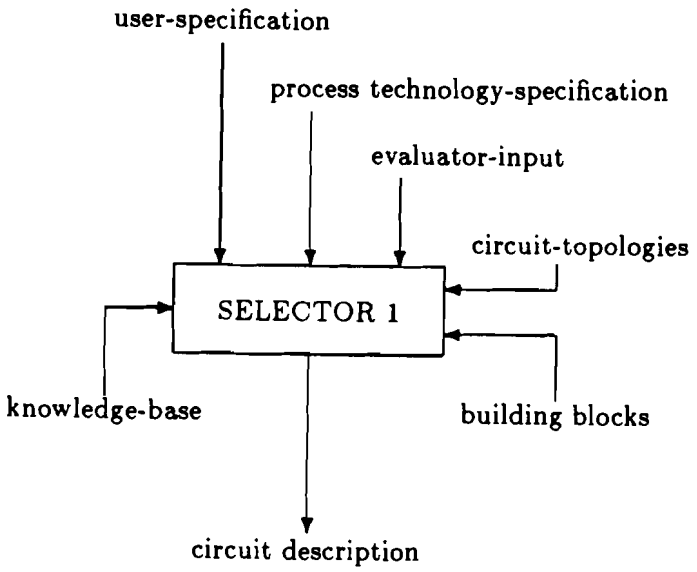



figure 3.4

**input:**

User specifications.
Process technology specifications.

**output:**

A circuit topology in terms of interconnected building blocks in short: a circuit description.

**KB-facts:**

Circuit topologies with relevant global formulas which estimate the possibilities of an architecture.

Building blocks with relevant global formulas which estimate the possibilities of a block.

**KB-rules:**

Design rules, rules of thumb.

**function:**

A topology is selected with which the interconnection pattern and the building blocks are fixed.
A building block performs a function. Depending on the specifications a building block can have a simple or a complex structure, an output block can consist of 1 or e.g. 10 transistors, thus having a more complex structure. If it has a more complex structure, the building block itself is constructed of sub-building blocks. So Selector-1 implicitly determines the complexity of a building block by selecting it.
Choices made are based on formulas and design rules.

### 3.2.2 Translator

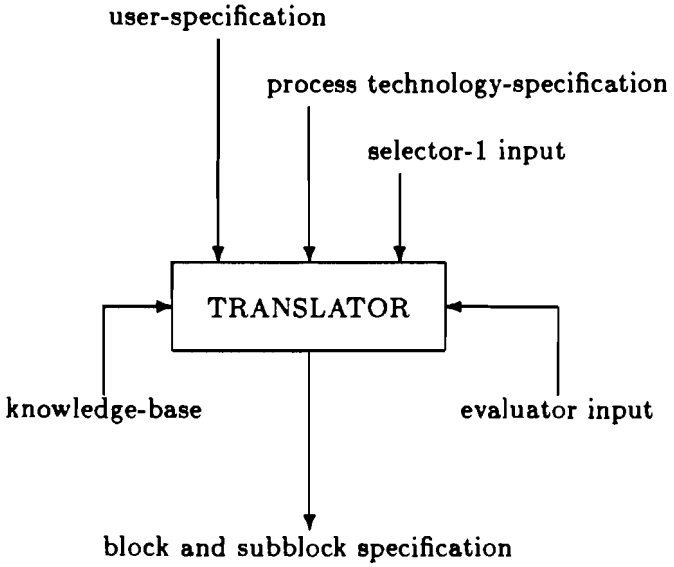



figure 3.5

**input:**

User specifications.
Process information.
A circuit topology.

**output:**

Specifications for every building block and specifications for every sub-building block.

**KB-rules:**

Rules to relate a block behaviour to parameters.
Rules to interpret the influence of a technology parameter on a building and sub-building block.
Rules to divide a parameter specification among building and sub-building blocks.

**function:**

To translate the circuit specifications towards building and sub-building block specifications by using rules and topology related formulas.

### 3.2.3 Planner

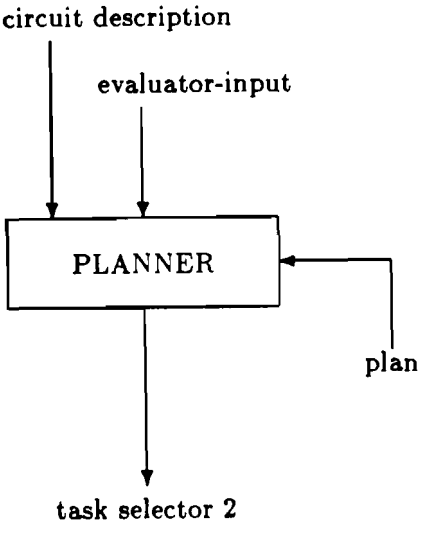



figure 3.6

**input :**

A circuit description.

**output :**

An instruction for selector-2 of where to start selecting and sizing the circuits, which perform the functionality of a building or sub-building block.

**KB-facts :**

A plan relevant to a topology and building blocks of how to attack the topology in order to synthesize the circuit.

**function:**

Managing the working order of selector 2.

### 3.2.4 Selector-2

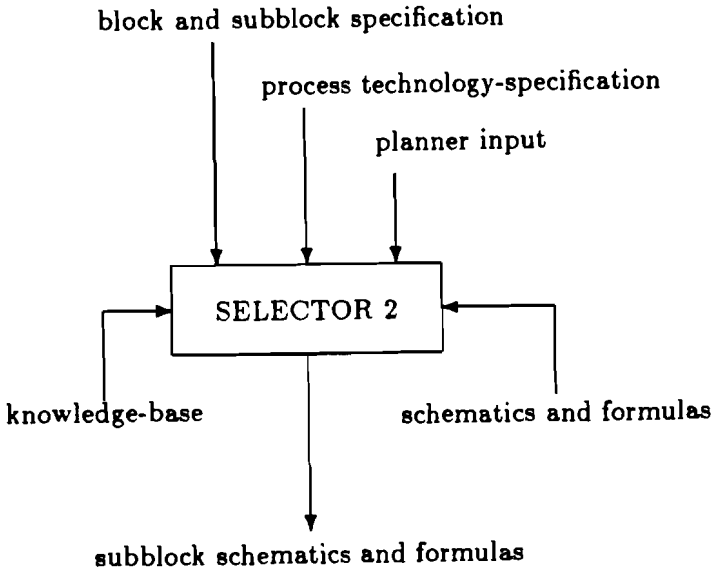



figure 3.7

**input :**

Technology information
Block specification from the translator.
Instructions from the planner.

**output :**

Circuit schematic with corresponding, relevant formulas.

**KB-facts :**

Basic circuit schematics with belonging formulas.

**KB-rules :**

Rules on how to select the best circuit according to the specifications.

**function:**

To select a transistorized schematic that can match the specifications.

### 3.2.5 Evaluator

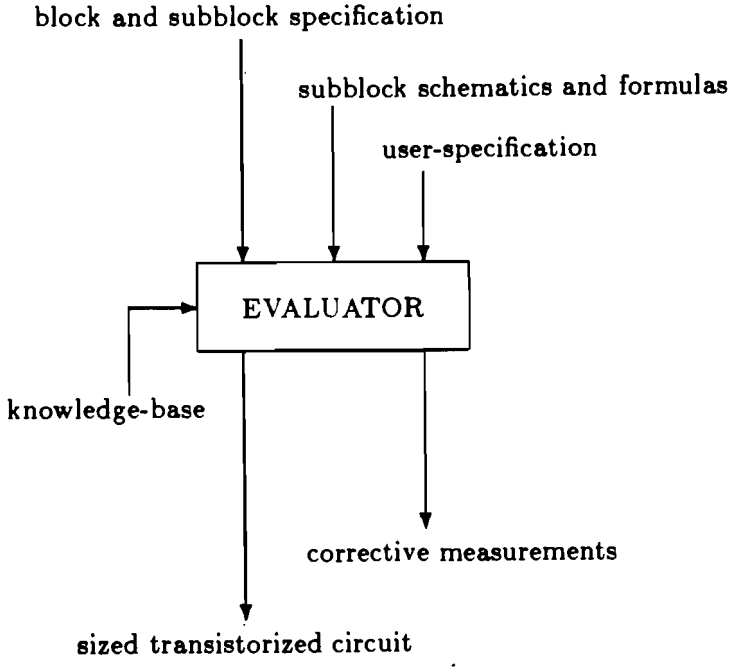



figure 3.8

**input :**

Circuit schematics with belonging formulas.
Translator specifications.
User specifications.

**output :**

Completed (sized) circuit schematics with component specifications according to the user specifications or a best possible schematic.
Corrective measurements.

**KB-rules :**

Analytical rules concerning the interpretation of the results.
Rules on how to react on the outcome of the analysis.

**function:**

To solve the equations according the specifications.
When it turns out that it is not possible to match a specification, give instructions:

- to select another circuit (by selector-2)
- to adjust the specification (by the translator)
- to select a complete new topology (by selector-1)

This is what is meant by corrective measurements.

## 3.3 Evaluation of the proposed framework

This proposal was discussed with and by my attendants and accepted. A problem was signalled on what to do about DC problems. Choices made, based on high-frequency characteristics, can conflict with DC demands.
The request to minimize the time needed to introduce a new design into the systems data (knowledge) base can be honored by structuring the circuit knowledge with help of the system. In this phase it was suggested that perhaps a kind of formula generator could be of help but that was found not necessary.
No solution was found to solve the problem of the adjustability of the system towards rigorous technology changes. For the moment the system was relatively independent towards technology changes by using a separate technology input.

Realizing the complexity of the whole project it was necessary to concentrate on a task with which the principal functioning of the system could be tested and that was surveyable. Doing a redesign was perfectly suited for this. Talking in terms of the system it meant to design and build first: the translator, the planner and the evaluator.
Before starting to do this I had to find out what kind and to what extent a designer was able to give information about a newly finished design. As a test two designers were interviewed about a design they had made, according to the following information model.

1. Generate a nodal description of the design.
2. Which parameter set characterizes the circuit sufficiently.
3. Divide the circuit into signalpath components and none signalpath components.
4. Distinguish current-voltage sources, reference sources, levelshifters and protection circuitry.
5. Distinguish the remaining parts of the circuit as separate building blocks.
6. Mark transistor devices that have to be identical, e.g. the transistors in a differential pair.
   Mark transistors that differ only a factor in length or width in respect to each other, e.g. from a reference current source derived current sources.
7. Define the operational area of every transistor.
8. Give simplified formulas describing the circuit behaviour in terms of circuit parameters defined by component parameters.
   With these formulas it must be possible to estimate the circuit range.
9. Give a connection between specification and component parameters in formula form.
10. Give a criterion of how to divide the realisation of a specification to parts of the circuit (e.g. gain, to be realized in input and output stage, factorized as 1 to 5).
11. Describe how to approach the circuit when sizing the components.

12. Give rules of thumb to initialize the search towards a solution.

During the interview several problems occurred. First it was disputed if and how to partition the circuit into separate blocks. It was not always possible to tell unambiguously to which block a certain component should be attributed because it had a major influence on the behaviour of more blocks and without including it in the blocks, it was not realistic to distinguish blocks. So the question arose if it was necessary to do so since the complete circuit behaviour could be captured in a few formulas. The latter turned out to be an even more serious problem.
To describe a circuit parameter in terms of component parameters was not as easy as expected. It was very difficult for a designer to give detailed formulas because during the design the only formulas used are simplified formulas. Of the simplified formulas one could question the reliability because in a new design not all the effects of a component on the circuit behaviour are completely understood. Therefore component influences are sometimes wrongly related with a certain effect on a parameter resulting in wrong formulas.
According to the formulas, let there be no doubt that an expert designer can thoroughly describe a circuit by formulas. The only question is how long it would take to do so. Experience with SCOPA has taught that for a relatively simple circuit it took about two years to describe it in such a way that dependable redesigns could be made. The people who created SCOPA will never want to do such a terrible job again. Furthermore SCOPA will become relatively worthless when the process technology used is further developed or changed and certain neglects in device models are not allowed to be made anymore.
Returning to the question of whether partioning a design in separate blocks is necessary. Because of the archive function it is important to do this in case of reference sources and other clearly distinct and reusable basic circuits. For doing redesigns its important to identify identical parts, e.g. much common in ladder structures, to facilitate the resizing of components. Otherwise it is not really necessary to do so other than with the trivial parts like current sources. The formulas are sufficient.
This thesis only holds if we can generate the formulas which describes every parameter related with component parameters. Of course the Kirchoff-laws derived DC-formulas are not the problem. The AC-small signal domain formulas are. Because of reasons described before and summarized below these formulas could not be given by a designer.

- Deriving formulas is a tedious and time consuming job.

- When a designer derives formulas he uses simplified models for the transistors. The use of these simplified models is only allowed if the assumptions made, according to the models, can be verified.

- Component influences are sometimes wrongly related with a certain effect on a parameter resulting in wrong formulas.

A routine was needed with which a formula could be generated. Instead of continuing the development of the synthesis tool, I started with a not yet available analysis tool. Sansen at the university of Leuven is developing such a tool (see also the next chapter). After finishing this task I would return my attention towards synthesis.

# Chapter 4

# Formula, a formula generator

## 4.1 Introduction

Formula is meant to derive circuit behaviour describing formulas (equations) which could not be given by a designer. Formula is doing the same job and is based on the same method as part of a program under development by Sansen at the University of Leuven, Belgium [C-30]. The idea of writing a routine to derive formulas existed before I new of the work from Sansen who, as it turned out, also had a contract with Philips concerning further development of the program. I proceeded with my own routine because the program was not yet available.
Formula is based on nodal linear network analysis by use of a nodal admittance matrix. A description of this method can be found in every handbook concerning network analysis or basic lecture notes on electronics e.g. [B-23,26]. I will give here only a short description of how to construct a nodal admittance matrix.

## 4.2 Direct construction of a nodal admittance matrix

A nodal admittance matrix is constructed by proceeding through the following steps:

1. The circuit components are, if necessary, replaced by a component model[1] that consists of current sources, resistors and capacitors.

[1] Remark: It is also possible to use norators and nullators to describe a component model but the program is not implemented to use such a model so I will not mention it further.

2. The circuit nodal points are numbered 1 to n where the lowest numbers are reserved for first the input nodes and second the output node. The numbering order of the remaining nodes is free.

3. The connection between two nodes is called a branch which can consist of multiple elements. There are two types of branches:

Type-1 The admittance branch.
The typical branch is directed from node m to n with an admittance Yr as a two-terminal element (figure 4.1). Now Yr is placed in the matrix on the following positions:
+ Yr on positions MM and NN
- Yr on positions MN and NM

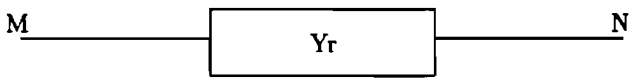



figure 4.1

Type-2 The voltage controlled current branch.
A typical branch is directed from node i to j containing a voltage controlled current source (figure 4.2). The controlling voltage is connected between node k l.
Now the transconductance factor Gm, is placed in the matrix on the following positions:
+ Gm on positions IK and JL
- Gm on positions IL and JK

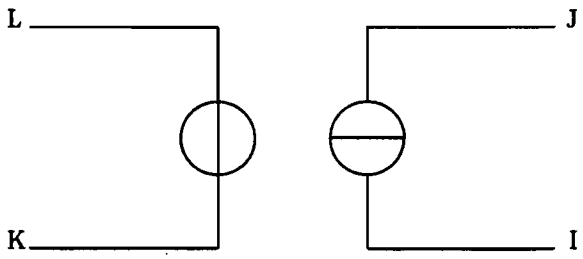



figure 4.2

The sum of all elements in a row and the sum of all elements in a column must be zero. In figure 4.3 an example is shown.

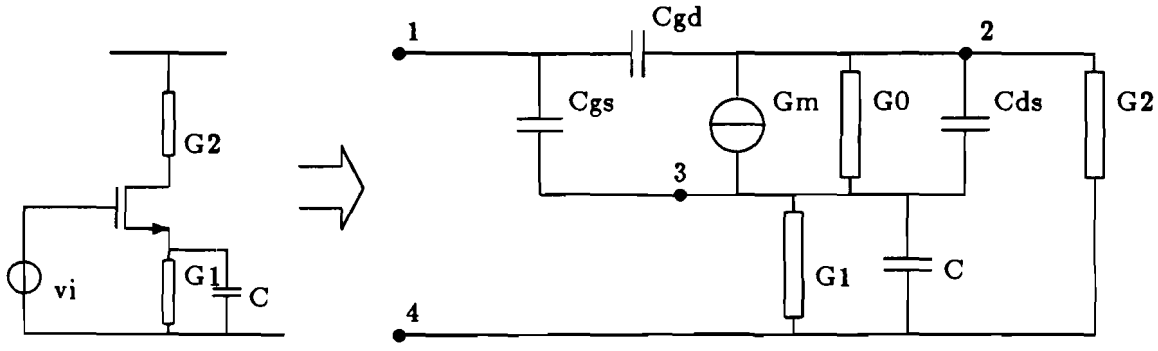



figure 4.3a

$$\begin{pmatrix} jw(Cgs+Cgd) & -jwCgd & -jwCgs & 0 \\ Gm-jwCgd & G0+G2+ \\ & jw(Cds+Cgd) & -G0-Gm-jwCds & -G2 \\ -Gm-jwCgs & -G0-jwCds & G0+G1+Gm+ \\ & & jw(Cgs+Cds+C) & -G1-jwC \\ 0 & -G2 & -G1-jwC & G1+G2+jwC \end{pmatrix}$$

figure 4.3b

The items listed in the array of figure 4.3b are admittances.

A circuit is viewed at as a two-port network, (figure 4.4), with one common reference.

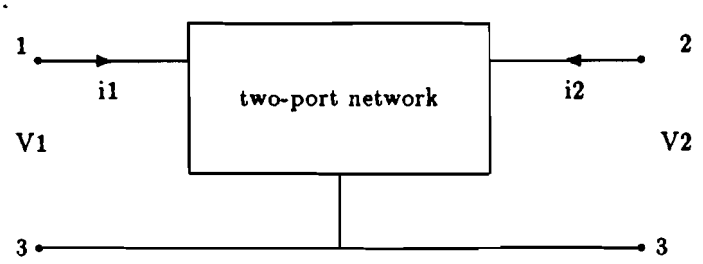



figure 4.4

The relation between voltages and currents is given by:

$$\begin{pmatrix} i1 \\ i2 \end{pmatrix} = \begin{pmatrix} Y11 & Y12 \\ Y21 & Y22 \end{pmatrix} \begin{pmatrix} V1 \\ V2 \end{pmatrix}$$

In general we have got an n-port network which we have to reduce to a two-port network. In order to do this, the matrix is ordered in input node on row and column 1 and output node on row and column 2. A reference node is selected (node 3 in figure 4.4) and the belonging row and column number can be deleted (see also method description in literature). We now transform the n-1 by n-1 matrix into a 2 by 2 matrix by repeatedly using Cramers rule. As an example lets use a 5 by 5 matrix that we transform in a 2 by 2 matrix.

$$\left(\begin{array}{cc:ccc} Y11 & Y12 & Y13 & Y14 & Y15 \\ \hdashline Y21 & Y22 & Y23 & Y24 & Y25 \\ Y31 & Y32 & Y33 & Y34 & Y35 \\ Y41 & Y42 & Y43 & Y44 & Y45 \\ \hdashline Y51 & Y52 & Y53 & Y54 & Y55 \end{array}\right) \Rightarrow \begin{pmatrix} \overline{Y11} & \overline{Y12} \\ \overline{Y21} & \overline{Y22} \end{pmatrix}$$

The transformation to for instance $\overline{Y12}$ is done as follows:

$$\overline{Y12} = \frac{\begin{vmatrix} Y12 & Y13 & Y14 & Y15 \\ Y32 & Y33 & Y34 & Y35 \\ Y42 & Y43 & Y44 & Y45 \\ Y52 & Y53 & Y54 & Y55 \end{vmatrix}}{\begin{vmatrix} Y33 & Y34 & Y35 \\ Y43 & Y44 & Y45 \\ Y53 & Y54 & Y55 \end{vmatrix}}$$

The denominator is the determinant of the matrix formed by the indices of the to be reduced nodal points. The numerator is the determinant of the matrix formed by the indices of the to be reduced nodal points, the element whose new value we want to account and the row and column above and beside the to be reduced nodal points.
For $\overline{Y12}$ in the original matrix, dashed boxes are drawn arround the elements that form the numerator.
The same is done for $\overline{Y11}$, $\overline{Y21}$ and $\overline{Y22}$. In general: let the original matrix describe n nodal-points. The to be reduced nodal-points are the nodes k to n. The matrix formed by these nodes is indicated by $A_{k,n}$. Every new element $\overline{Y_{i,j}}$ is then accounted by:

$$\overline{Y_{i,j}} = \frac{\begin{vmatrix} Y_{i,j} & Y_{i,k} & - - - & Y_{i,n} \\ Y_{k,j} & & & \\ \vdots & & A_{k,n} & \\ Y_{n,j} & & & \end{vmatrix}}{|A_{k,n}|}$$

Out of the 2 by 2 matrix we can derive several circuit characteristics like gain, input-impedance, output-impedance, CMRR etc. because they are simple relations of e.g. Vin-Vout or Vin-Iin. With this method, the influence of components and component-parameters on circuit characteristics is accounted.

## 4.3 Results

After implementing the theory I experimented with several circuits and then it became very clear why a designer uses very simplified models and even then will think twice before starting to derive a formula. I mention the results of the $\frac{Vout}{Vin}$ transfer function of two simple opamp circuits as an illustrative example. The opamp of figure 4.5 contains 5 different transistor devices and for the transfer function generates 7 pages (A4) of output. The opamp of figure 4.6 contains 6 different transitor devices and generates more than 60 pages (A4) of output. We are talking about just one equation! In trying to generate the transfer function for an 8 different transistor devices complex circuit, the 50 Mbyte memory space of the VAX was to small to generate the formula.

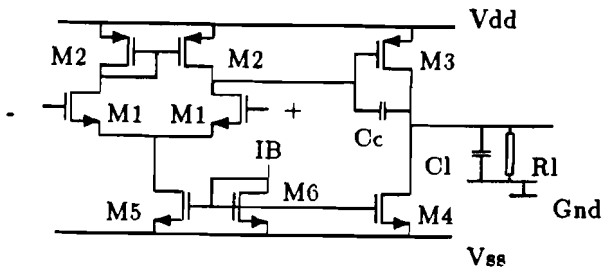



figure 4.5, Miller opamp

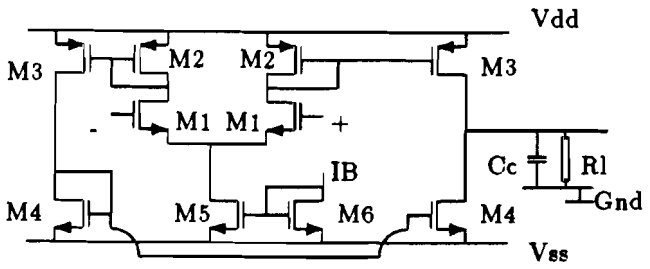



figure 4.6, OTA

Not only the amount of memory space but also the cpu-time needed to generate a formula demanded optimisation of the program on which much time was spent. Optimising the program however, did not yet solve the problem of the gigantic output. To reduce the formula length two methods have been implemented:

- A possibility to reduce the model to simplify the expression.
  Parts of the model can be neglected, for every transistor or for any individual transistor in the circuit.

- A possibility to reduce the expression by numerical valuation.
  The expression is compounded of $jw^n$ fractions. A fraction consists of a symbolic sum of products. A value is attached to every symbol which indicates the magnitude of the symbol. There is made no distinction between components. Every parasitic capacitor, every capacitor, every admittance, every transconductance is valuated the same ($Gm = 10^{-3}, CGs = 10^{-13}$ etc).
  Of a fraction, the maximum product contribution is accounted. All contributions smaller than a user chosen percentage of the maximum are deleted. The percentage can be set to every real number equal or smaller than 100%. Depending on the percentage chosen, the reduction is substantial. In appendix-A, the results of the transfer function for the circuits in figure 4.5 are listed in a table. Also in appendix-A an example of a 10% reduced transfer equation of the circuit in figure 4.5 is given.

  There are two possibilities for reducing the expression by valuation:

  - after the expression is completed
  - during the formatting of the expression

  When the second option is chosen an error is introduced since identical products, when valuated seperately, are deleted but summed would be taken into account. Symbolic summation however is done only after the whole expression is formatted. During the experiments on five different circuits it turned out that this error is small for a factor above 10, less then 5% of a fractions total value (worst case experimental result).

Use of the second option is sometimes a necessity because the virtual memory is too small. The program will generate a warning in this case.

## 4.4 Conclusions

Formula is very useful for deriving formulas. A user is able to reduce expressions according to his own wishes. Four different ways for deriving expressions have been implemented but this was only done to experiment. With a better user-interface, the user should be able to define any formula derivation and is limited only by the possibilities to recognize a formula in a nodal admittance matrix structure.
Although formula was in the first instance meant to derive formulas used for redesigning a circuit, it can also be useful in a design process by a designer to check the influence of a component on a parameter. As an example I recall the experience of an expert designer during a demonstration of the similar program of Sansen when a component had influence on a parameter. The expert, who was certain of the fact that it could not have any influence, was wrong!

One major drawback is the enormous output generated by the program. A problem that perhaps can be solved by a different way of formatting the formula. There was no time available to investigate this possibility.
Another possible solution I suggest is found in the formula structure. For large circuits, there are many (n) nodes and components, also the amount of jw fractions is up to n. If the frequency domain in which a circuit is meant to function is taken into account, the contribution of the highest jw fractions can be neglected thereby reducing the formula. This of course also depends on what information one is looking for. As an example:

$$| \, gain \, | = | \, 1000000 + (jw * 0.0001) \, |$$

If the maximum frequency is 10MHz then the jw-factor can be neglected. In using such knowledge when deriving formulas for large ciruits, not only the output is reduced but also the CPU-time needed to account the formula is reduced because not everything need to be accounted for. The option to set a limit to the amount of jw fractions will be implemented.

## 4.5 Program description

The program consists of 16 functions, whose functionality I will explain in brief.

- Formula
  Formula is the main body of the program and acts more or less as a user-interface. The user is questioned on what equation is to be derived, if the model used for the transistor component has to be simplified, if reduction by numerical valuation is required, if the result has to be saved.
  Formula then invokes the appropriate functions to derive the desired equation.

- admittance-matrix-generator(AMG):
  The AMG takes a spice-like inputfile, an example is shown in appendix-A, and places the components into a nodal admittance matrix as explained in this chapter. Therefore AMG uses the functions

  - NMOS
  - PMOS
  - resistor
  - capacitor

  These functions contain the model definition of a component and when called, return the model-components and the positions of their entries in the nodal admittance matrix.
  Furthermore, AMG generates an INFO (INFOrmation) list which contains information about which nodal numbers where used in the circuit description, how many components of a kind are encountered, to what nodes voltage and current sources are connected.
  INFO has the following format:

```
( (nodal-numbers)
  (+DC-node -DC-node GND-node dc-current-node +AC-node -AC-node)
  ( (Transistor-indices)
    (Resistor-indices)
    (Capacitor-indices))
  (nodal list of nodes used in the sparse matrix))
```

The dc-current-node is the connection node between a symbolic current source and the circuitry.
The information in the last entry of INFO is added by the make-sparse function. The information in INFO is used throughout the formula and synthesis program.

- make-sparse (MS)
  MS transforms an array in to a sorted sparse matrix. The matrix is sorted in Vin-node, V-out-node and other nodes. The DC-nodes and the reference node are deleted. The result is a nodal admittance matrix of the AC-small-signal circuit schematic.
  The nodes that construct the sparse matrix are added to the INFO list.

- N-matrix, Prob
  N-matrix generates the determinant or sub-determinant of a matrix. Prob is used to call N-matrix from the main program. A separate call function is used because N-matrix invokes itself repeatedly with different initial variables.

- Popper, Layer-reductor (LR), Reduce-and-sort (RAS)
  These functions write out the nested sum and multiplier expression.

$$((m * a * b * (c + d)) + (n * n))$$

$$\Downarrow$$

$$((m * a * b * c) + (m * a * b * d) + (n * n))$$

  Popper does this by recursively calling itself every time a new list is encountered, writing out this new list first. For large expressions control stack problems occur.
  To overcome this problem LR was written. LR performs, in writing out an expression, the same function as popper. Furthermore, LR separates the elements in positive and negative jw-fractions by use of the RAS function. Also a reduce-list is included. The elements in this list are neglected by LR, when writing out the nested expression.
  LR first determines how many nested layers the expression has, then LR starts writing out the two inmost layers and from there outward, a layer at a time.
  LR uses popper to write out a layer. LR uses RAS to sort the layer

elements in positive and negative elements and to separate them in jw-fractions.
Elements in the reduce-list are deleted thereby reducing the expression.

If requested, LR invokes the numerical-reductor routine, to further reduction of the expression.

- Numerical-reductor (NR), Call-num
  As explained in section 1.3, every symbol in the list is given a value after which the maximum of a jw-fraction is calculated. Depending on the percentage chosen, elements within a certain percentage of the maximum, are deleted from the expression.
  Call-num is used to invoke the numerical reductor.

- Symbolic-reductor (SR)
  SR collects identical symbol or symbol combinations to reduce the length of the expression.

$$((a * b) + (a * b)) \Longrightarrow (a * b * 2)$$

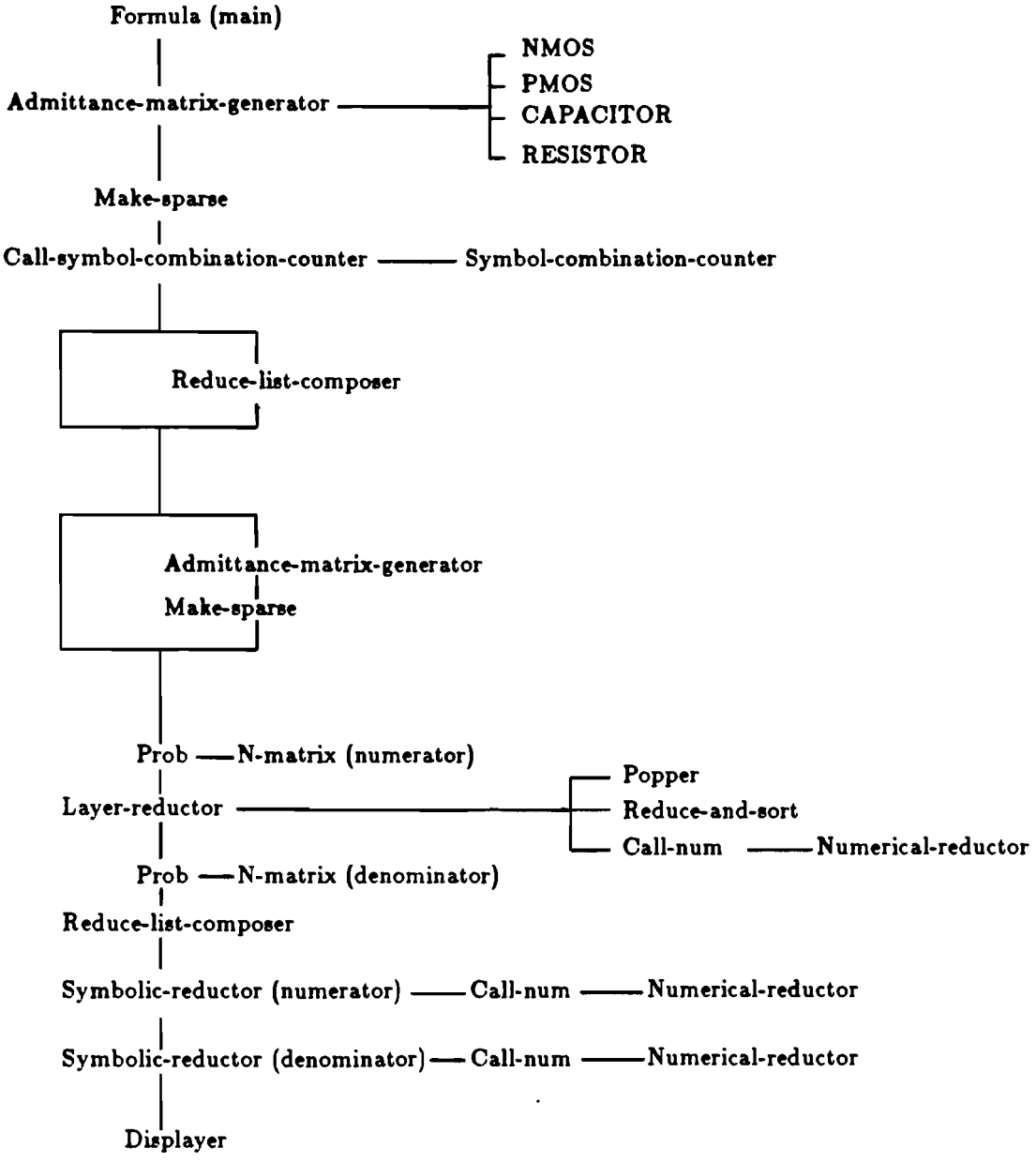
  SR does this by using hash-tables. The expression can be further reduced by the numerical-reductor.

- Displayer
  Displayer formats the output (an example is shown in the appendix-A) of the formula program. The length of the expression is calculated and the user is asked if the expression has to be saved in a file in the directory and/or shown on the screen.

- Symbol-combination-counter (SCC), Call-symbol-combination-counter (CSCC)
  SCC is used to estimate the amount of elements that can be generated by formula. The outcome is an indication of whether the virtual memory space available, is sufficient to account the formula. CSCC is used to invoke SCC.

To gain some insight into how the functions are called inside Formula, see figure 4.7, where a call-chart of Formula is drawn.

figure 4.7, Call-chart Formula



# Chapter 5

# Synthesis

## 5.1 Introduction

Synthesizing a circuit means firstly taking a lot of decisions based on global rules and formulas concerning

- which circuit topology to select,
- which building blocks to use,
- which transistorized circuit is the best

after which finally, the sizing of the components can take place. Due to the complexity of analog synthesis I had confined myself to redesign. This means that the complete circuit schematic of a design is known and no selections have to be made concerning topology, building blocks and transistor configurations. The problem is therefore reduced to sizing the components of a known circuit.
This sizing of components is not as easy as it might seem in first place. True, we have the circuitry with its corresponding formulas but we also have a set of parameter specifications which can conflict with each other, as is most often the case. We have a set of formulas that has to be solved in such a way that the set of specifications are satisfied.
The sizing problem is in fact equivalent to finding a global minimum in an n-dimensional space where the dimensions are formed by the component parameters. There are several ways to attack this problem. Which to choose strongly depends on how the description of a circuit by equations is organised and how complex they are. Another important factor is which

parameter or parameters are awarded with the most value.
A few examples are:

- An iterative procedure like the Newton Raphson method or one of the many other gradient methods.
- Accounting everything by a step by step procedure where a route is laid out along which to proceed towards a solution.
- To direct the search towards a solution in an interactive way by letting the user adjust specifications or revalue the importance of a parameter. The Delight Spice system works this way [A-13].
  In that case, the system can also point towards conflict situations or give some sort of advise.

In general there exists no solution for the problem of finding a global solution in a n-dimensional space. Whatever method is chosen and in spite of ones certainty that a solution exists, there is no guarantee that the solution is found.

## 5.2 Synthesis by use of a cost function

In this case there was a set of complex parameter equations that had to be solved. I wanted to solve these equations by creating a cost-function of them, which then had to be minimized.
The cost-function is of the form:

$$cost = \sum_{i=1}^{n} (weight_i * (parameter_i - specification_i))^2$$

where n is the number of parameters and weigh,t a factor used to stress the importance of a parameter. The optimal solution is found if the cost is or approaches zero. To find the optimal solution the cost-function had to be minimized. Minimizing the cost-function meant to solve a combination of equations. Combined these equations had, in the case of an eight transistor circuit, approximately 40 variables.
Fortunately these variables were not completely independent. The model used was based on a Philips transistor model called the "MOS-model-7" [C-31] which was implemented in a program called MOSCA [C-34]. Experiments with the model (by use of the program) showed that with a chosen DC

set up, the variance of all the model components was, in a certain range, approximately linear with the transconductance factor. This meant that the number of variables of the cost-function could be reduced considerably . This is done by the following:

1. For every transistor, the value of the model components for a certain DC set up were accounted while the transconductance variable ($Gm$) is set to a chosen value, for example $26 * 10^{-6}$.

2. Every model component belonging to a certain transistor is substituted by its value times a factor symbol, e.g. $Gm1$ is set to $26 * 10^{-6} * F_1$ and $Cgs1$ is set to $18 * 10^{-15} * F_1$ etcetera. The values can be calculated by MOSCA according to a chosen voltage Vds and a chosen size of L of the transistor.
   As an example let
   $Gm1 = 1 * 10^{-6}, Gm2 = 2 * 10^{-6}, Cgs1 = 1 * 10^{-15}, Cgs2 = 3 * 10^{-15}$
   and A, an arbitrary equation:

   $$A = (Gm1 * Cgs2 * Gm2) + (Gm1 * Gm1 * Cgs1)$$

   After substitution it is transformed to:

   $$A = (F_1 * F_2^2 * 6 * 10^{-27}) + F_1^3 * 10^{-27}$$

3. The range by which linearity approximation could be used is also determined using MOSCA, with Gm set to $26 * 10^{-6}$ the range of F is: $0.1 < F < 10$. This range is not arbitrary but depends strongly on the correctness of the relations between Gm and the other components.

   Remark!!!... The dependency of the model-components of Gm is strongly related with the operational point of the transistor. Because we have to reckon with the DC adjustment of the circuit, this relationship is seldom linear and also different for every transistor. (See also the test-results section.)

What was used now was a numerical optimization routine that could handle wild functions with several variables and minimize them. Such method was not available in the standard VAX library.
After doing some experiments with gradient methods on solving the equations, I decided not to use them because of the complexity of the equations involved.

A general method which could be used to solve the problem is the Simplex Method of Nelder and Mead(1965) [B-25] not to be mistaken with the Simplex Method for linear optimization. A general description of this method can be found in any standard handbook on optimization such as Walsh(1975) or Wolfe(1978). It was a pity they were not available for me so I was very lucky to receive a description of this method next to the short introduction in [B-25] of one of my colleagues. It turned out that there were some differences in how the method could be applied. Therefore and because of the difficulty one will presumably have in finding a description of the specific method implemented, a complete description of the method is given.

## 5.3 The Simplex Method of Nelder and Mead

### 5.3.1 General introduction to the Simplex method

In this subsection the basic principles of the Simplex method are described. In the next subsection a description of the Nelder and Mead variant implemented, is given.

- Define: $\xi_1, ...., \xi_m \epsilon \Re^n$, $m \leq n$ linear independent.
  With $X^0 \epsilon \Re^n$ and $X^i = X^0 + \xi_i, i = 1, .., m$ then

  $$S(X^0, X^1, .., X^m) = \{X \mid X = \sum_{i=0}^{m} \lambda_i X^i, \sum_{i=0}^{m} = 1, \lambda_i \geq 0\}$$

  is called a m-Simplex in $\Re^n$.
  $X^i, i = 0, .., m$ is called a Simplex-corner. The connection between two corners is called a flank , see figure 5.1. If all flanks of a Simplex are equal then the Simplex is called regular.

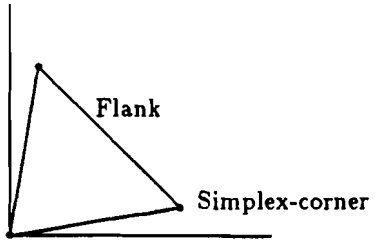



figure 5.1, regular 2-Simplex in a two dimensional space

- Account for every Simplex-corner the function-value $f_i = f(X^i)$ and determine $X_g$ to be the largest $f_i$. Then
  Define: $X_{cent} = \frac{1}{n}(\sum_{i=0}^{m} X_i - X_g)$, called the centroid of the Simplex (figure 5.2).

- Define: $S(X^0, X^1, .., X^m) \subset \Re^n$ is a n-Simplex then

$$S_{reflect}(X^0, .., X^{i-1}, 2X^i_{refl} - X^i, X^{i+1}, .., X^n)$$

  is called the reflected Simplex.
  The transformation from

$$S(X^0, X^1, .., X^n) \Rightarrow S_{reflect}(X^0, .., X^{i-1}, 2X^i_{refl} - X^i, X^{i+1}, .., X^n)$$

  is called the reflection step from $X_i \equiv X_g$.

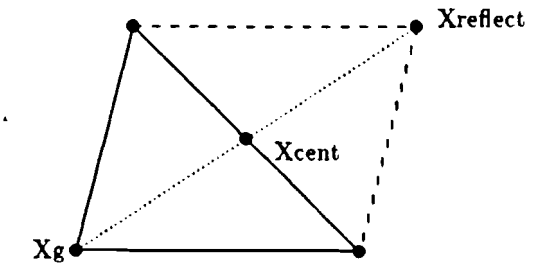



figure 5.2

- Define: The transformation from

$$S(X^0, X^1, .., X^n) \Rightarrow S_{con/exp}(\overline{X}^0, \overline{X}^1, .., \overline{X}^n)$$

  with $\overline{X}^i, \overline{X}j = X^i + \alpha(X^j - X^i)$ is called:

  - a *contraction* if $0 < \alpha < 1$
  - an *expansion* if $\alpha > 1$

  obtained from $X^i \equiv X_j$. These contraction and expansion steps are uniform (figure 5.3).

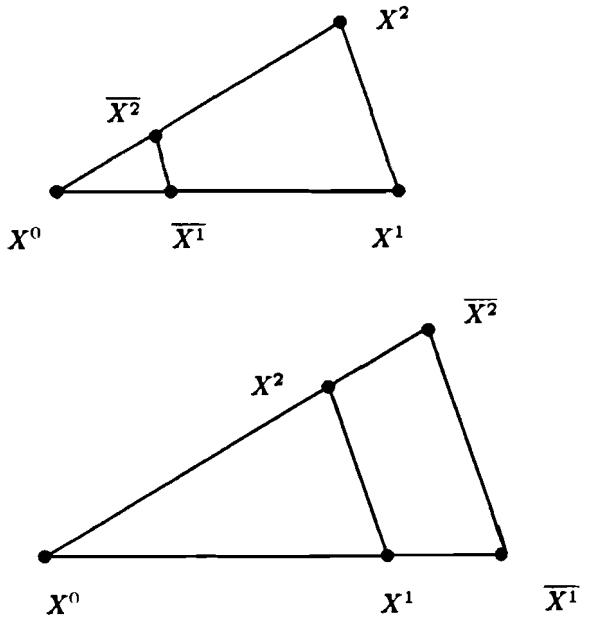



figure 5.3

The Simplex-method tries, by employing a variety of alternative moves (reflection, contraction and expansion) on a Simplex, to find a minimum.

### 5.3.2 The Nelder and Mead method

The difference between the Simplex-method as described before and the Nelder and Mead variety is found in the facts that next to the uniform contraction and expansion steps, these steps can be made in several directions and non-uniform, depending on the circumstances.
The transformation from

$$S(X^0, X^1, .., X^n) \Rightarrow S^{i,\alpha}(X^0, X^1, .., X^n) = S(X^0, .., X^{i-1}, \overline{X^i}, X^{i+1}, .., X^n)$$

is done by $\overline{X^i} = X^i_{cent} + \alpha(X^i_{cent} - X^i)$.
In case $\alpha = 1$: the transformation is a reflection, shown in figure 5.4a
In case $0 <| \alpha |< 1$: the transformation is a contraction. There are two posibilities for contracting the simplex. This is shown in figure 5.4b and c.
In case $| \alpha |> 1$: the transformation is an expansion, shown in figure 5.4d.

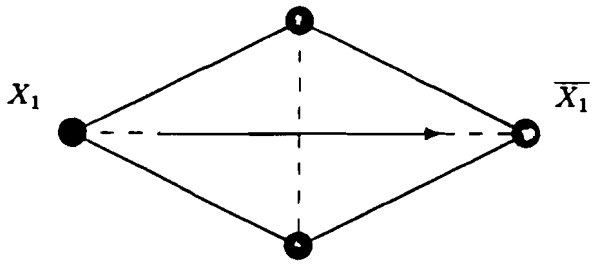



figure 5.4a, $\alpha = 1$

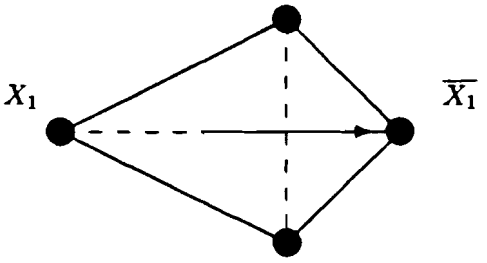



figure 5.4b, $0 < \alpha < 1$

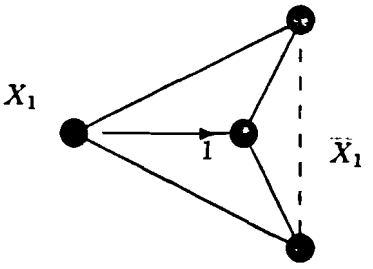



figure 5.4c, $-1 < \alpha < 0$

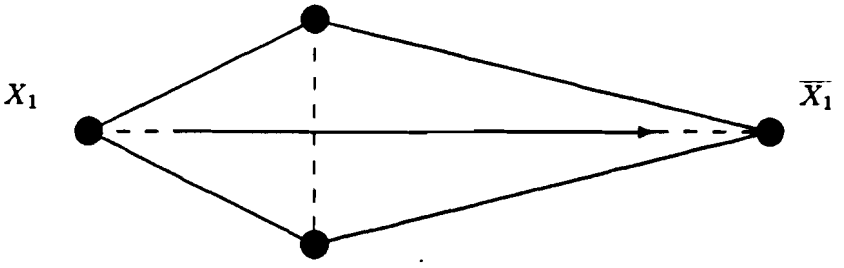



figure 5.4d, $| \alpha | > 1$

### 5.3.3 Start and stop conditionals

It is advised to start with a regular simplex. The standard construction of such simplex with flank-length 1 is shown. The initial point chosen here is (0 0 .. 0 0).
We place the Simplex-corners in an n by (n+1) matrix

$$\begin{pmatrix} 0 & \alpha & \beta & .. & \longrightarrow & \beta \\ 0 & \beta & \alpha & .. & \longrightarrow & \beta \\ \downarrow & .. & .. & .. & \longrightarrow & \beta \\ \downarrow & .. & .. & .. & \searrow & \beta \\ 0 & \beta & \beta & \longrightarrow & \longrightarrow & \alpha \end{pmatrix}$$

To find $\alpha$ and $\beta$ we have to solve the next two equations:

derived from flank 1 (figure 5.5): $\alpha^2 + (n-1)\beta^2 = 1$

derived from flank 2 (figure 5.5): $2(\alpha - \beta)^2 = 1$

resulting in:

$$\alpha = \tfrac{1}{n\sqrt{2}}(n - 1 + \sqrt{(n+1)})$$

$$\beta = \tfrac{1}{n\sqrt{2}}(\sqrt{(n+1)} - 1)$$

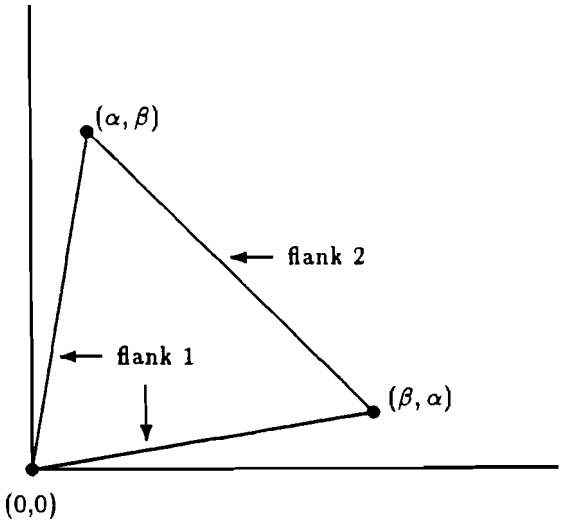



figure 5.5

In general when we want to construct a Simplex with flank length $\delta$:

$$\alpha = \frac{1}{n\sqrt{2}}(n - 1 + \sqrt{(n+1)})\delta$$

$$\beta = \frac{1}{n\sqrt{2}}(\sqrt{(n+1)} - 1)\delta$$

So $X^i = X^0 + (\beta\beta..\alpha..\beta)$ ($\alpha$ on i-th position), i = 1,....,n .

The expansion factor $\alpha_{exp}$ is recommended to be 2.
The contraction factor $\alpha_{con}$ is recommended to be 0.5 .

The stop criterium is given by the normal-deviation:

$$\sigma^2 = \frac{1}{(n+1)} \sum_{i-0}^{n} (f_i - \overline{f})^2$$

$$\overline{f} = \frac{1}{(n+1)} \sum_{i=0}^{n} F_i$$

where $\sigma < \varepsilon$.
The iteration is stopped when the Simplex has become too small.
A second stop criterium is of course, to limit the amount of iterations.

### 5.3.4 Conditions and actions

Start with the start Simplex as described above and repeat the following steps:

- determine $f_i = f(X^i)$ and sort resulting in: $f_n \geq f_{n-1} \geq .... \geq f_0$
- reflect $S(X^0, X^1, .., X^n)$ in $X^n \equiv X_g$, $\overline{X}^n = 2X^{cent} - X^n$; $\overline{f} = f(\overline{X}^n)$
- if $\overline{f} < f_0$ then $\overline{\overline{X}}^n = X^n_{cent} + \alpha_{exp}(X^n_{cent} - X^n)$; account $\overline{\overline{f}} = f(\overline{\overline{X}}^n)$
  - if $\overline{\overline{f}} < f_0$ then $X^0 := \overline{\overline{X}}^n$; $f_0 := \overline{\overline{f}}$
  - if $\overline{\overline{f}} \geq f_0$ then $X^0 := \overline{\overline{X}}^n$; $f_0 := \overline{\overline{f}}$
- if $\overline{f} \leq f_{n-1}$ then $X^n := \overline{X}^n$; $f_n := \overline{f}$
- if $f_{n-1} < \overline{f} \leq f_n$ then $\overline{\overline{X}}^n = X^n_{cent} - \alpha_{con}(X^n_{cent} - X^n)$

  else $\overline{\overline{X}}^n = X^n_{cent} + \alpha_{con}(X^n_{cent} - X^n)$

account $\overline{\overline{f}} = f(\overline{\overline{X}}^n)$

If $\overline{\overline{f}} < f_n$ then $X^i := X^0 + \alpha_{con}(X^i - X^0), i = 1, ..n$

else $Xn := \overline{\overline{X}}^n; f_n := \overline{\overline{f}}$

This method was implemented and tested.

## 5.4 First results

The program was tested on the following function:

$$f(x, y) = (x - y)^2 + y^2$$

During the test the following observations were made:

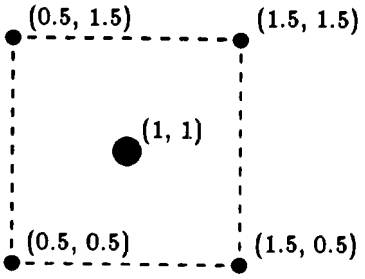
- When no restrictions were made concerning the range of the variables the minimum was found, independent of the initial-point. The ammount of iterations necessary was on average the same.
- When restricting the range, depending on the initial-point, problems occurred with finding the minimum.
  It was recognized that this was due to the limited degree of freedom for the expansion and contraction of the Simplex. The performance could be improved by varying the expansion and contraction factors.

In testing the method on the cost-function of a single transfer equation with a restricted variable range this problem was far more serious and the initial-point had to be a very good guess in order to find the global minimum. The enlargement of the problem was due to "wildness" of the function, more local minima, and the amount of variables. No method or approach was found that solved this problem completely.

## 5.5 Factor variation

Experiments showed, when a starting point was chosen near a solution, that there was a very good chance of finding the solution. This observation reduced the problem a bit towards finding a good initial-point.
The idea of factor variation is based on another observation. When choosing an arbitrary point it turned out that, after running the program when the

minimum was not found, the variables were into a certain proportion. This proportion could be expected although it strongly depends on the frequency domain. When the proportion was enlarged, it turned out that it could converge towards a solution.
This was translated into a routine that beginning with the initial-point (1 1 .. 1), simply tried out all the possible proportions and chose the best one to continue with the Simplex method.
As an example lets take the function $f(x, y)$ mentioned before. With the initial-point (1 1) the four possible proportions of x and y are:




| Proportion table | | |
|---|---|---|
| x | y | f(x,y) |
| 0.5 | 0.5 | 0.25 |
| 0.5 | 1.5 | 3.25 |
| 1.5 | 0.5 | 1.25 |
| 1.5 | 1.5 | 2.25 |

figure 5.6, Factor test points

The initial-point was (1 1). The proportion variation was $\pm 0.5$.
Looking at the results we can conclude that under constraint of minimizing f(x,y), we have to proceed with the (x,y) proportion (0.5 0.5).

## 5.6 Second results

After the implementation of Factor variation in combination with the Simplex method I repeated the same experiments as before thereby making the following observations:

- It was now possible to converge towards a solution by repeating Factor variation and the Simplex method with the result of one routine fed into the other.
- The result was still dependent on the choice of the initial-point. In the case where a solution exists, the worse the choice the more iterations were needed.
- It is best to start with Factor variation because using the Simplex method first can lead to the wrong variable proportion that can not

be overcome by Factor variation. In worst case it turned out that the two routines could come into a "deadlock" situation. This luckily could be solved by repeating the Factor variation several times or by changing the proportion variation.

- The best results were obtained with the initial-point (1 1 .. 1). In general it was best to start in the center of the variable domain.
- When using the Simplex routine to find a minimum I found that the routine would find a minimum in approximately 50 iterations. Doing more iterations showed not to be fruitful. Therefore the amount of iterations done by the simplex method is set to 50.

As an example in figure 5.7 and 5.8 the search for a global minimum for an arbitrary function with an x and y variable is shown. The variable range limits are indicated by the dashed lines. Figure 5.7 shows the points found in search for a global minimum by use of the combination of the Simplex and the Factor method. Starting at point 1 with Factor variation results in a new best point at 2. At point 2 the Simplex method finds a new best point at 3 after which the Factor method is repeated. At point 8 the Simplex method does not find a better point, the Factor method comes up with point 9. Finally at 10 a solution is found. In figure 5.8 the search for a global minimum is shown when the Simplex method can not find a better point after the Factor variation.

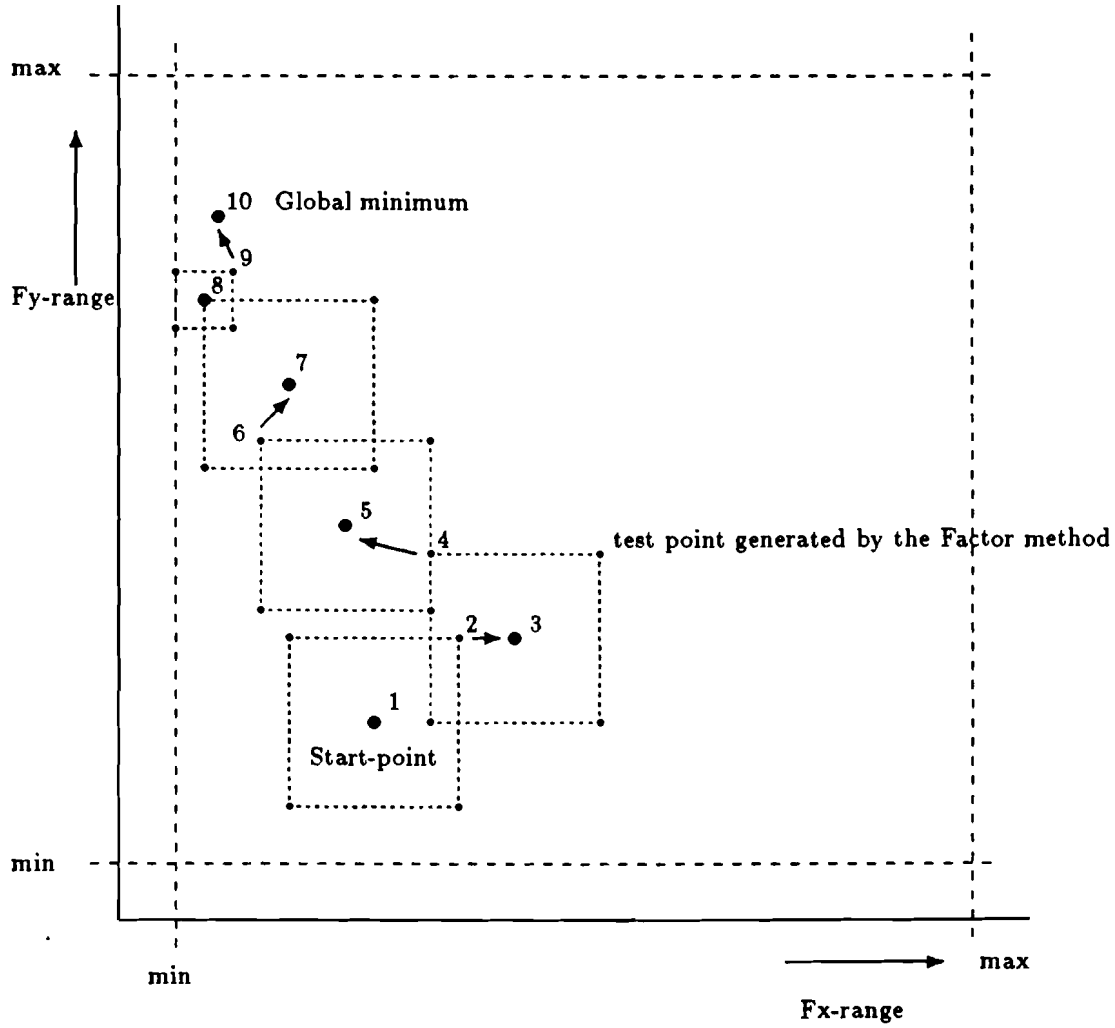



figure 5.7, minimum search by combining the Factor and the Simplex method, an arrow indicates a point found by the Simplex method

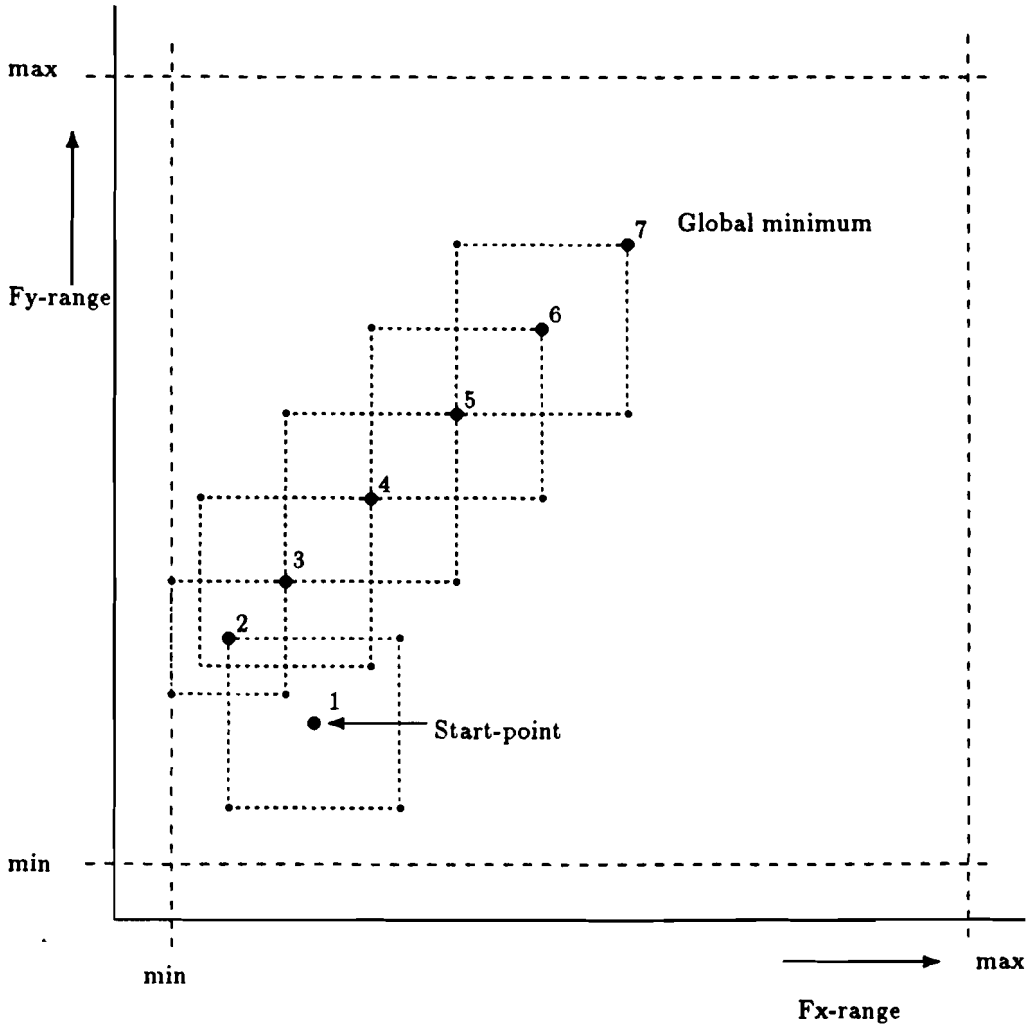



figure 5.8, minimum search when the Simplex method finds no better best points

## 5.7 Test results

Tests done so far, only showed that it was possible to find a solution with using both methods discussed. Untill now, only the AC-part of the circuit was looked at. When synthesizing a circuit, also the DC-part had to be taken in to account. To do this, the following strategy was used:

1. Choose for every transistor a channel length L
2. Choose a DC-operational point for the circuit, and determine for every transistor:

   - Vgs
   - VT
   - Vds
   - Vbs
   - Ids

3. Determine the variation range according to the variation of for instance: Gm, VT, Vgs or Ids etc.
4. Determine the initial values of the model-components using MOSCA and determine a correction factor for Cgs, Cgd, Cdb and G0 in relation with the variation of Gm
5. Determine the factor-range for Gm
6. Construct the cost-function
7. Minimize the cost-function
8. Check the DC operational points and if necessary adjust them.
   In the latter case, account for the new operational points of the transistors the component-model values, reformat the cost-function and check the result. If the result is not satisfactory, repeat all steps starting with step 3.

Choosing a strategy was relatively easy, but to employ it was more troublesome. Being directed towards the AC-domain of circuit design, no routines were written to control the DC-domain. Since the MOS-model-7 was implemented in the program called MOSCA, also no routine was written to calculate the necessary transistor-model-components. This meant that after

every iteration, the DC-values, the model-components, the determination of the new range and the correction-factors had to be calculated by hand (of course using MOSCA). It turned out that this was almost impossible and no circuit was synthesized this way.
It is absolute necessary to include the circuit-DC-operation-point function and the relation between the transistor-DC-operation-point and the transistor-model-components in the cost-function. This relation is found in the MOS-model-7 There was not enough time left to do this so this is one of the jobs of which my successor has to take care.
In order to test the reliabilty of the used formulas, the DC voltages and currents were determined with Philpac, the model-components were derived by using MOSCA and the transfer function and the phase were calculated by philpac and by using the formulas. The result is shown in in appendix-B. The discrepancy between the formula result and the Philpac result is due to the fact that in Philpac the complete MOS-model-7 is implemented while the model used by Formula was only a limited version of the MOS-model-7. The result is good enough to regard the derived formulas as reliable.
This result is sufficient to conclude that, with the circuit-DC-operation-point function and the relation between the transistor-DC-operation-point and the transistor-model-components included in the cost-function, at least redesign is possible. However, before this conclusion can be drawn definitely, more experimentation will be necessary.

## 5.8 Conclusion and remarks

The first results regarding the finding of a solution for a set of equations and a set (the maximum used uptil now during experiments was four) of conflicting parameter specifications are promising, since good solutions for different parameter constraints were found. Some remarks have to be made relating to this statement:

- No guarantee exists that a global solution will be found even if it is there in principle.

- If a specification for a circuit parameter is on the bounds of what is possible with the circuit, the more difficult it is to find a solution. The best results are then gained by repeatedly doing the Factor variation (figure 5.8).

- Factor variation is a very arithmetic intensive job which can easily get out of hand. In finding an initial solution one should reduce the amount of variables as much as possible.

- Again, as with the fomula routine, Factor variation can be used, during the design process by a designer, to get an impression of what influence a certain component has on a parameter or a set of parameters.

The problem of redesigning a circuit seems to be solved. Because of lack of time, the DC-adjustment and the neccessary MOS-model-7 formulas could not be implemented and therefore it was not possible to indeed synthesize a circuit. The DC-adjustment however, is not viewed as being a problem. (Adjusting the DC-levels after every iteration step is also done in SCOPA.) So it is expected that, when the DC-operation relations are included in the cost-function, redesigning of a circuit by using this tool is possible.

## 5.9 Program-description

The Simplex method of Nelder and Mead is implemented as already described before, in a separate function called SIMPLEX. To keep the simplex inside the allowed range, the movements of the simplex are limited where necessary. This is done by adjusting the expansion, the contraction or the reflection-factor dynamically, this before every movement.
The input consists, next to the input needed by the Simplex method, of the cost-function and a variable-list. The latter is generated by a function called SYMBOL-FINDER.
The cost-function consists of a list of lists:
cost-function = ((list-1)(list-2)....(list-n))
One such list makes up one parameter-specification part of the cost-function and consists of 5 parts:
(list-n) = (numerator denominator weight-factor specification test-criterion)
The test-criterion is used inside the simplex routine to test if a specification is met, e.g. parameter-equation result > than specification. If so then the contribution of this parameter-specification part to the cost-function is zero. The test criterion can be =, >, <, >= or =<. The cost-function can be generated by use of the SP22 function.
Also the variation-factor of the Factor variation method is adjusted in accordance with the allowed range. This method is implemented straightforwardly in a separate function called FACTOR.

Both methods are invoked repeatedly, from a small and very user-unfriendly routine called SYNTH, format:

(SYNTH (list of parameter-equation numerators)
(list of parameter-equation denominators)
(list of parameter specifications)
(list of parameter weights)
(list of fmin-range for every variable)
(list of fmax-range for every variable)

SYNTH uses SP22 to generate a cost function. The function SP22 is again a very user-unfriendly routine. SP22 asks for the values of every component symbol found in the cost-function and substitutes symbols by a factor and a value:
$(Gm1 * Gm2)$ is transformed to $(Fm1 * Gm1_{value} * Fm2 * Gm2_{value})$
SP22 also allows a correction-factor to be introduced but this is restricted to a factor, e.g. Fm1, or a value-factor combination, e.g.$(2 * Fm1)$. In the latter case
$(Gm1*Gm2)$ is transformed to $(Fm1*(2*Fm1*Gm1_{value})*Fm2*Gm2_{value})$.
This however has to be changed. (see also section test-results).
Of these functions only SIMPLEX and FACTOR are ready to use. The remainder are written to ease the testing.

# Chapter 6

# Conclusions

The purpose was to design a system with which reasonable designs could be made. A framework was presented with which this was thought to be possible. The framework was tested with and by expert designers and accepted. Because of the complexity of the problem of analog synthesis, I restricted myself to redesigns. In doing this, the principal functioning of the framework could be tested although for doing redesigns, only part of the proposed frame-work was needed, namely: the planner, the selector-2 and the evaluator (figure 4.3). In trying to gather the information needed, it occurred that for the evaluator, necessary and reliable equations could not be given by the designer.
To solve this problem, a formula-generator was build with which these equations could be derived. Although Formula performs well and is also useful independent of the synthesis system, problems still exist on how to handle large circuits. Perhaps this problem can be solved by Sansen at Leuven, who is working on a program with the same functionality.
The next problem that had to be solved, the sizing of the circuit, was how to find an optimum for a set of equations according to a set of conflicting parameter specifications. To solve this problem a cost-function was created which had to be minimized to find the optimum. The sizing problem was seen as equivalent to finding a solution in an n-dimensional space where the dimensions are formed by the component parameters. In the Simplex Method of Nelder and Mead, not to be mistaken with the Simplex Method for linear optimization, a general method was found with which this problem could be attacked. Because of the restricted range in which the variables could be varied, the chosen method turned out to be insufficient. The Factor-variation method was introduced and in combining this method with

the Simplex method, good results were reached.
As a final test an attempt was made to redesign a circuit. This could not be done because of a lack of routines, with which it was possible to adjust the DC-operation-point of the circuit and with which the MOS-model-7 components could be calculated directly.
It is expected that, when these routines are added, redesign will be possible.
Finally, testing of the derived formulas in accordance with the circuit simulator Philpac showed that the formulas were reliable.

Robin van Raaij, Eindhoven, augustus 1988.

# Chapter 7

# Literature-list

This literature-list is divided into:

- an article-list, in the report referred to by [A-number]
- a book-list, in the report referred to by [B-number]
- a company confidential-list, in the report referred to by [C-number].

The items on the company-list are not freely available.

## 7.1 Article-list


1. Foreward reasoning and dependency-directed backtracking in a system for computer aided circuit analysis.
   R.M. Stallman and G.J. Sussman
   Artificial Intelligence 9 pages 135-196, 1977

2. Adaptive biasing Cmos amplifiers.
   M.G. DeGrauwe, J. Rijmenants, E.Vittoz and H.J. de Man
   IEEE Journal of Solid State Circuits vol. sc-17 pages 522-528, june 1982

3. MOS operational amplifier design- A tutorial overview.
   P.R. Gray and R.G. Meyer
   IEEE Journal of Solid State Circuits vol. sc-17 pages 969-982, december 1982

4. Custom and semi-custom design techniques, a synthesis program for operational amplifiers.
M.G. DeGrauwe and W.M.C. Sansen
IEEE international Solid-State Circuits Conference pages 18-19, 1984

5. How circuits work.
J. De Kleer
Artificial Intelligence 24 pages 205-279, 1984

6. Blades: an expert system for analog circuit design.
F.M. El-turkey and R.A. Nordin
Proceedings of IEEE ISCAS pages 552-555, 1986

7. A knowledge based system for analog integrated circuit design.
R.J. Bowman and D.J. Lane
Proceedings of IEEE ICCAD pages 210-212, 1986

8. Intelligent system for analog design.
B.S. Cohen, J.L. Cuadrado and K. Kendall
IEEE 1986

9. A prototype framework for knowledge based circuit synthesis.
R. Harjani, R.A. Rutenbar and L.R. Carley
24th Design automation conference pages 42-49, 1987

10. An analog expert design system.
M. Degrauwe et. al.
IEEE ISSCC digest of technical papers pages 212-214, 1987

11. State of the art in the analog CMOS circuit design.
E. Habekotté, B. Hoefflinger, H.W. Klein and M.A. Beunder
Proceedings of the IEEE, vol 75 no 6, june 1987

12. Automatic synthesis of operational amplifiers based on analytical circuit models.
H.Y. Kok, C.H. Séquin and P.R. Gray
IEEE ICCAD pages 502-505, 1987

13. Delight Spice: an optimization system for design of integrated circuits.
B. Nye, J. Spoto and A. Tits
IEEE 1988

14. Expert system-based operational amplifier design.
    A.H. Fung, P.Y. Pun, Y.N. Lai and B.J. Sheu
    Department of Electrical Engineering and Information Science Institute, University of Southern California, Los Angeles, CA 90089

15. A fully automated expert system design environment for operational amplifiers.
    F.M. El-Turkey

16. Computer assisted "cut and try" design of the logic and the layout of combinatorial circuits.
    P. Szolgay

## 7.2 Book-list

17. Building expert systems
    edited by F.H. Roth, D.A. Waterman and D.B. Lenat

18. The MYCIN experiments of the Stanford heuristic programming project
    B.G. Buchanan, E.H. Shortliffe

19. Integration of analogue electronic circuits
    J. Davidse

20. Lisp, a gentle introduction to symbolic computation
    D.S. Touretzky

21. Lisp
    P.H. Winston and B.K.P. Horn

22. Common Lisp, The language
    G.L. Steele jr.

23. Computer aided analysis of electronic circuits: algorithm and computational techniques
    L.O. Chua and P.M. Lin

24. Introduction to numerical analysis
    J. Stoer, R. Bulirsch

25. Ledermann Handbook of applicable mathematics, Volume 3, Numerical methods
editor R.F. Churchhouse

26. Elektronica 1 part 1 & 2
lecture notes course no: 5C04.0
J.H. van den Boorn, G.G. Persoon,
Department of Electrical Engineering Technical University Eindhoven (TUE)

27. Moderne elektronica lecture notes course no: 5L030
K. Breukers, J.A.W. Faatz,
Department of Electrical Engineering Technical University Eindhoven (TUE)

## 7.3 Company-list

28. Cursus inleiding IC-ontwerp, analoge basisschakelingen (2e correctie)
A. Sempel

29. VERA, VERification Assistant
Functional specification version 1 draft 2, 15 june 1987
Drs. A.P. Kostelijk and Ir. G.G. Schrooten
Philips research laboratories

30. Voorstel project ESAT-Philips

31. MOST model-7, PURPLE handbook, april 1987

32. Technical note no: 179/85
Analysis of two-stage Miller opamps with high-ohmic output
A.H.M. van Roermund

33. Technical note no: 220/86
Design of Miller opamps for switched capacitor applications
A.H.M. van Roermund

34. Technical note no: 222/85
MOSCA, A computer program for "MOSCA"lculations
A.H.M. van Roermund, P.H. Seesink

35. Technical note no: 2618
Sensible cicuit help for COCO
M.B. Burchell

36. Technical note no: 2544
Study of feasability of Artificial Intelligence (AI)
Techniques to the control and observation of circuit optimisation (COCO)

M.B. Burchell, A.D. Smith

37. Technical note no: 6139
SCOPA, a Switched Capacitor OPAmp design program
A.H. van Roermund, P.H. Seesink

38. User's guide IDAC (Interactive Design of Analog Cicruits)
version 2.3, 2 november 1987
Centre Suisse d'Electronique et de Microtechnique S.A. (CSEM)

# Appendix-A

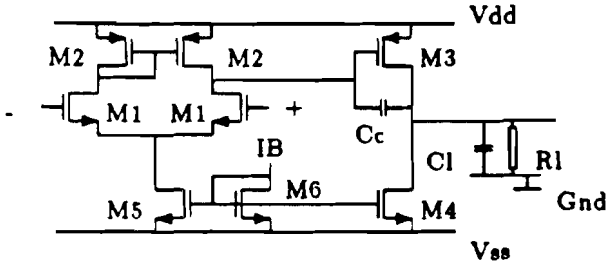



figure 4.5, Miller opamp

The circuit-input file is:

```
(setq name '"
(D G S B
*
M1 5 1 7 0 NMOS
M1 6 2 7 0 NMOS
M2 5 5 8 0 NMOS
M2 6 5 8 0 NMOS
M3 3 6 8 0 NMOS
M4 3 9 10 0 NMOS
M5 7 9 10 0 NMOS
M6 9 9 10 0 NMOS
VDD 8 0 DC 5
VSS 10 0 DC -5
VIN 2 1 AC 1
IB 9 0
CC 6 3 PF
CL 3 0 PF
RL 3 0 K)")
```

| Miller, figure 4.5 | | | |
|---|---|---|---|
| factor | CPU-time | Real-time | elements |
| 1000 | 12.8 | 19.91 | 969 |
| 999 | 12.57 | 17.86 | 795 |
| 100 | 9.47 | 10.97 | 332 |
| 99 | 9.55 | 11.26 | 185 |
| 10 | 6.37 | 7.36 | 117 |
| 9 | 5.86 | 8.94 | 94 |
| 1 | 5.79 | 8.45 | 17 |
| | seconds | seconds | |

Factor 1000 means that every element $< \frac{maximum-element}{1000}$ is deleted.

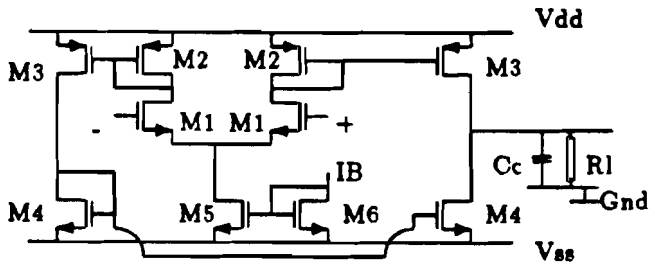



figure 4.6, OTA

| OTA, figure 4.6 | | | |
|---|---|---|---|
| factor | CPU-time | Real-time | elements |
| 1000 | 169.26 | 238.73 | 9153 |
| 999 | 166.83 | 238.73 | 7836 |
| 100 | 104.18 | 198.57 | 4125 |
| 99 | 104.85 | 198.57 | 3365 |
| 10 | 62.43 | 165.13 | 1766 |
| 9 | 48.65 | 69.01 | 1685 |
| 1 | 47.93 | 72.31 | 61 |
| | seconds | seconds | |

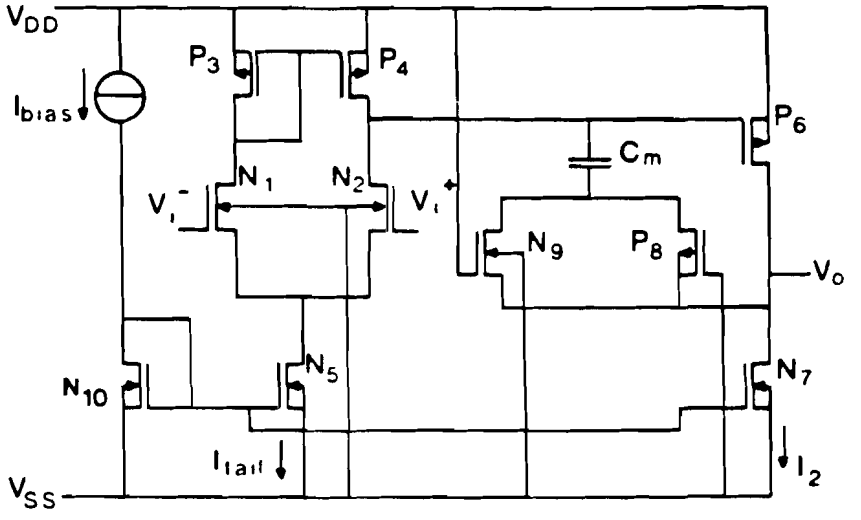
An element has the following form: (* A B C D)
The transfer equation of Miller with factor 10 is listed below:

```
The numerator is:

The coefficient of P**0 is:
(* GM1 GM1 GM2 GM3 2)

The coefficient of P**1 is:
(* -1 (* CC GM1 GM1 GM2 2))

The coefficient of P**2 is:
(- (+ (* CC GM1 GM2 CGD1 2) (* CC GO1 GM2 CGD1 2) (* CC GM1 GO2 CGD1 2))
   (+ (* CC GO1 GM1 CGS2 2) (* CC GM1 GM2 CDS5) (* CC GM1 GM1 CGS2 2)
      (* CC GM1 GM2 CGD5) (* CC GM1 GO5 CGS2 2) (* CC GM1 GM1 CDS2)
      (* CC GM1 GM2 CGS1 2) (* CC GM1 GM1 CGD1) (* CC GM1 GM2 CDS1 2)))

The coefficient of P**3 is:
(- (+ (* CC GM1 CGD1 CGD1 2) (* CC GM1 CGD1 CGS2 4) (* CC GM1 CGD1 CGD2 2)
      (* CC GM1 CGD1 CDS2 2) (* CC GM2 CGD1 CGD5) (* CC GM2 CGD1 CGS1 2)
      (* CC GM2 CGD1 CDS5) (* CC GM2 CGD1 CDS1 2))
   (+ (* CC GM1 CGD1 CGD5) (* CC GM1 CGS2 CDS5 2) (* CC GM1 CGD1 CGS1)
      (* CC GM1 CGD1 CDS5) (* CC GM1 CDS1 CDS2) (* CC GM1 CGS1 CGS2 2)
      (* CC GM1 CGS2 CGD5 2) (* CC GM1 CDS1 CGS1) (* CC GM1 CGS1 CDS2)
      (* CC GM1 CDS1 CGS2 2) (* CC GM1 CDS2 CDS5) (* CC GM1 CDS2 CGD5)
      (* CC GM1 CDS1 CDS5) (* CC GM1 CGD2 CGD5) (* CC GM1 CGD2 CDS5)
      (* CC GM1 CDS1 CGD5)))


The coefficient of P**4 is:
(+ (* CC CDS1 CGS1 CGD2 2) (* CC CDS1 CGS1 CGS2 2) (* CC CGD1 CGS1 CDS2 2)
   (* CC CGD1 CGD1 CGD5) (* CC CGD1 CDS1 CGS1 3) (* CC CGD1 CDS1 CGS2 4)
   (* CC CGD1 CGD2 CDS5) (* CC CGD1 CGS1 CGS2 4) (* CC CGD1 CDS1 CGD2 2)
   (* CC CGD1 CDS1 CGD5) (* CC CGD1 CGS1 CGD2 2) (* CC CGD1 CDS1 CDS1)
   (* CC CGD1 CDS1 CDS2 2) (* CC CGD1 CDS1 CDS5) (* CC CGD1 CGD1 CDS1 2)
   (* CC CGD1 CGD2 CGD5) (* CC CGD1 CGS2 CDS5 2) (* CC CDS1 CDS1 CGS1)
   (* CC CGD1 CGD1 CDS5) (* CC CGD1 CGD1 CGS1 2) (* CC CGD1 CDS2 CGD5)
   (* CC CDS1 CGS1 CDS2) (* CC CGD1 CGS2 CGD5 2) (* CC CGD1 CDS2 CDS5))
```

```
The denominator is:

The coefficient of P**0 is:
(+ (* GL GO1 GM1 GO2 2) (* GM1 GO2 GM2 GO3 2) (* GL GO1 GO2 GM2 2)
   (* GL GO1 GM1 GM2 2) (* GM1 GO2 GM2 GO4 2) (* GO1 GM1 GM2 GO3 2)
   (* GL GM1 GO2 GO2 2) (* GO1 GM1 GM2 GO4 2) (* GL GM1 GO2 GM2 2)
   (* GL GO1 GO1 GM2 2))

The coefficient of P**1 is:
(+ (* CC GL GM1 GM2 2) (* CC GM1 GO2 GM3 2) (* CC GM1 GM2 GM3 2)
   (* CC GM1 GO2 GM2 2) (* CL GM1 GO2 GM2 2) (* CC GM1 GM2 GO4 2)
   (* CC GM1 GM2 GO3 2) (* CC GL GM1 GO2 2) (* CC GL GO1 GM2 2))

The coefficient of P**2 is:
(+ (* CC CL GM1 GM2 2) (* CC CL GM1 GO2 2) (* CC CL GO1 GM2 2))

The coefficient of P**3 is:
(+ (* CC CL GM2 CGS1 2) (* CC CL GM1 CDS1) (* CC CL GM1 CGD2 2)
   (* CC CL GM2 CGD5) (* CC CL GM1 CDS2 2) (* CC CL GO1 CGS2 4)
   (* CC CL GM1 CGD1 2) (* CC CL GM1 CGS2 4) (* CC CL GM2 CDS1 2)
   (* CC CL GM2 CDS5))

The coefficient of P**4 is:
(+ (* CC CL CDS1 CGS2 4) (* CC CL CGD2 CGD5) (* CC CL CDS2 CDS5)
   (* CC CL CDS2 CGD5) (* CC CL CGD2 CDS5) (* CC CL CGD1 CGD5)
   (* CC CL CDS1 CDS5) (* CC CL CGD1 CGS1 2) (* CC CL CGS1 CGD2 2)
   (* CC CL CGD1 CDS5) (* CC CL CDS1 CGD5) (* CC CL CGS2 CDS5 2)
   (* CC CL CDS1 CDS1) (* CC CL CGD1 CDS1 2) (* CC CL CDS1 CDS2 2)
   (* CC CL CGS1 CGS2 4) (* CC CL CDS1 CGD2 2) (* CC CL CGS2 CGD5 2)
   (* CC CL CDS1 CGS1 2) (* CC CL CGS1 CDS2 2))
```

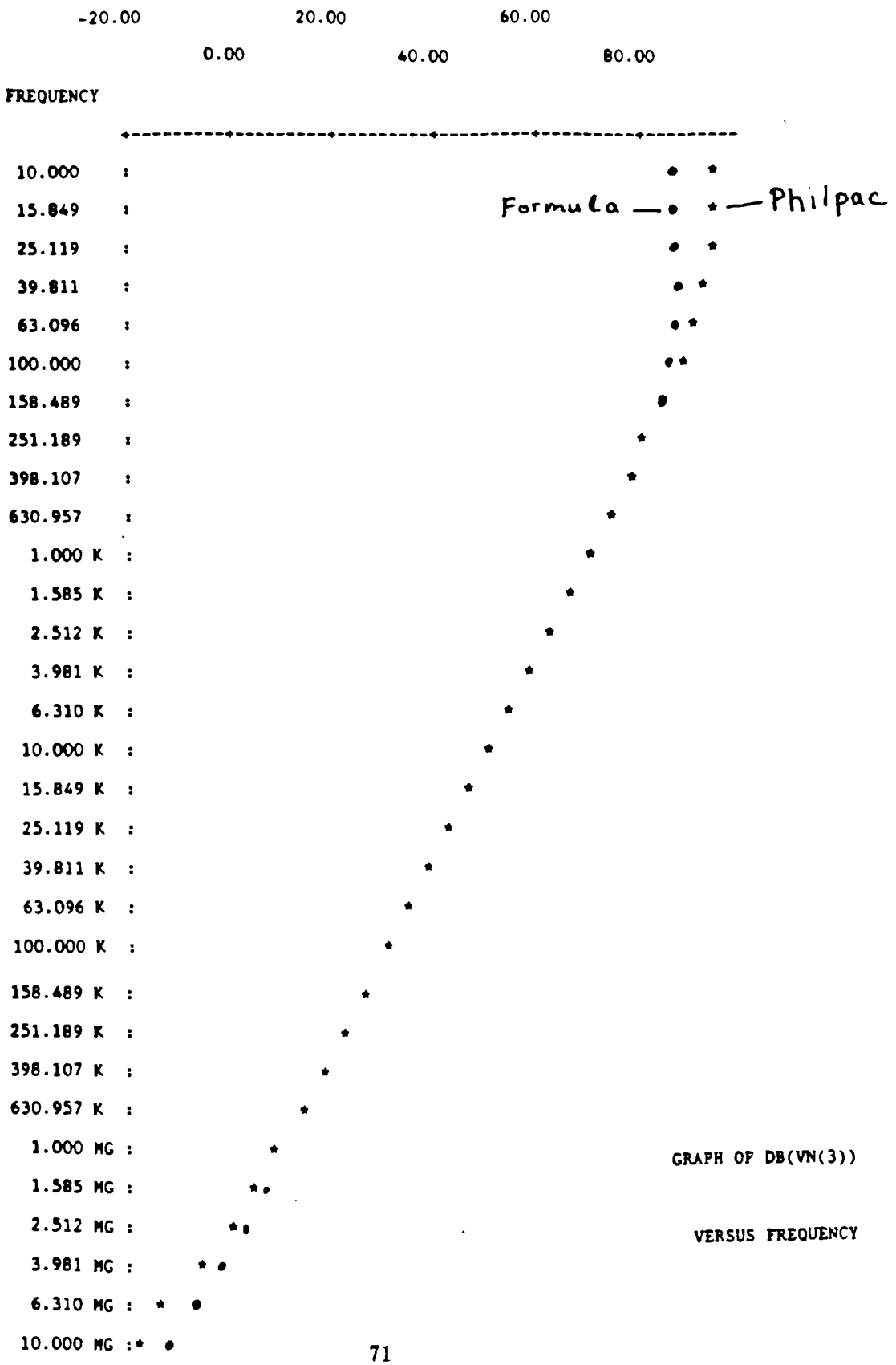
# Appendix-B

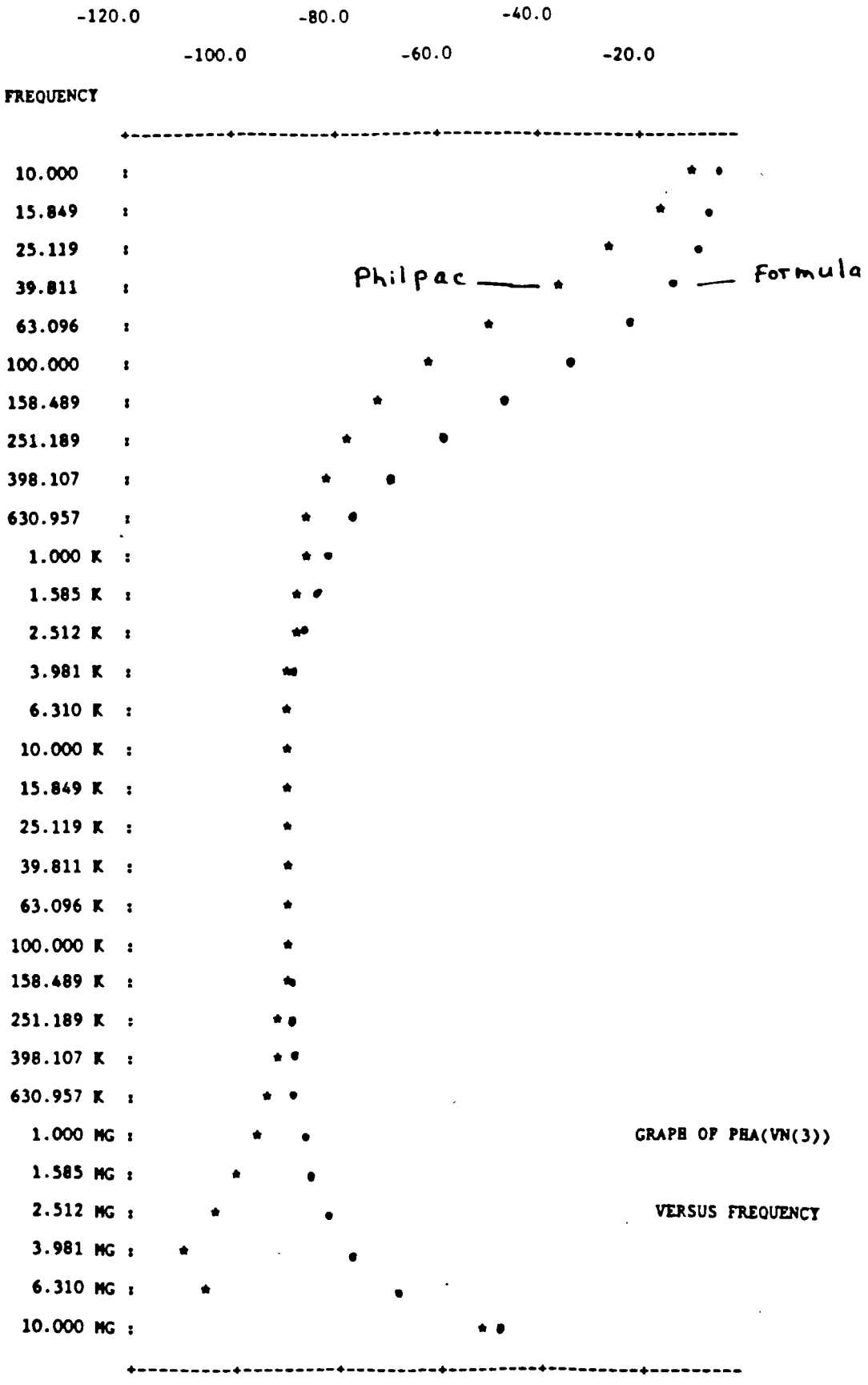
Formula test results compared with the Philpac results:




Circuit diagram of a 'NIPO' opamp (N-well process).

```
J1(10,6)6e-6 $
NTPA1(4,0,5,9),WU=5.6,LU=6.6,MULT=1 $
NTPA2(7,1,5,9),WU=5.6,LU=6.6,MULT=1 $
PTPA3(4,4,10,10),WU=10.4,LU=6.6,MULT=1 $
PTPA4(7,4,10,10),WU=10.4,LU=6.6,MULT=1 $
NTPA5(5,6,9,9),WU=3,LU=6.6,MULT=1 $
PTPA6(3,7,10,10),WU=19.4,LU=6.6,MULT=1 $
NTPA7(3,6,9,9),WU=3,LU=6.6,MULT=1 $
PTPA8(8,9,3,3),WU=3.2,LU=10,MULT=1 $
NTPA9(8,10,3,9),WU=5.4,LU=40,MULT=1 $
NTPA10(6,6,9,9),WU=3,LU=6.6,MULT=1 $
C1(7,8)2.6e-12 $
C2(3,0)128e-15 $
END $
AC $
f=GN(10,10MG,30) $
PR:DBPHA(VN(3)) $
PL:DBPHA(VN(3))S
endS
Run $
```


-20.00
0.00
20.00
40.00
60.00
80.00
FREQUENCY
10.000
15.849
25.119
39.811
63.096
100.000
158.489
251.189
398.107
630.957
1.000 K
1.585 K
2.512 K
3.981 K
6.310 K
10.000 K
15.849 K
25.119 K
39.811 K
63.096 K
100.000 K
158.489 K
251.189 K
398.107 K
630.957 K
1.000 MG
1.585 MG
2.512 MG
3.981 MG
6.310 MG
10.000 MG
Formula — Philpac
GRAPH OF DB(VN(3))
VERSUS FREQUENCY


-120.0
-80.0
-40.0
-100.0
-60.0
-20.0
FREQUENCY
10.000
15.849
25.119
39.811
63.096
100.000
158.489
251.189
398.107
630.957
1.000 K
1.585 K
2.512 K
3.981 K
6.310 K
10.000 K
15.849 K
25.119 K
39.811 K
63.096 K
100.000 K
158.489 K
251.189 K
398.107 K
630.957 K
1.000 MG
1.585 MG
2.512 MG
3.981 MG
6.310 MG
10.000 MG
Philpac
Formula
GRAPH OF PHA(VN(3))
VERSUS FREQUENCY