CAD Tool for Amplifier Design on Negative-Feedback Nullor Based Amplifiers

by

M.C. Roberto Castañeda Sheissa

Thesis submitted to the department of Electronics of the Instituto Nacional de Astrofísica, Óptica y Electrónica in partial fulfillment of the requeriments for the degree of
Doctor of Philosophy

November, 2007
Tonantzintla, Puebla

Principal adviser:

Dr. Arturo Sarmiento Reyes
INAOE

Summary

Traditional circuit design has as starting point a similar already proved design that could accomplish the desired specs. Once the circuit is selected, modifications are performed within it in order to provide the required performance. Unfortunately, this way to perform circuit design has several drawbacks, among them there are two such as:

- The time needed to perform the adaptation of the circuit is too high.
- Given the case that there is no circuit that could be adapted, it is necessary to invest valuable time and resources to create a new one.

This thesis presents the design and development of an automated tool aimed at the design of negative feedback amplifiers based on the *Structured Circuit Design* theory. The main idea of this methodology is to establish an ideal solution first and from this starting point, perform the required modifications to find the particular solution of the problem. The base element of this methodology is the *nullor*, which is an ideal two-port element with these important characteristics:

- The voltage and current at the input port always are zero.
- At the output port, the voltage and current values will always accomplish the load requirements.

This methodology allows to create amplifiers with one or two feedback loops and the topologies that can be implemented are: voltage amplifier, transconductance amplifier, transimpedance amplifier and current amplifier.

To reach the implementation of the amplifier is necessary, at first instance, to perform the nullor synthesis. The synthesis is performed by means of active devices since they are the only ones capable to emulate the nullor characteristics. Nevertheless, the active devices have some limitations like: noise generation, limited operational frequency or a narrow bias range. The active devices in this work have been modelled using their *small signal models*, this way is possible to depict any active device using just capacitors, resistors and controlled sources. Using the small signal models are valid through the design process because the main goal is to obtain a Butterworth like output, accomplish certain noise level and the output signal are kept within certain voltage or current level. Therefore, the main focus of this work lies in the AC domain while the DC bias values will be taken into account just in certain design aspects.

The design methodology allows to divide the problem into smaller, and easier to handle, problems that represent the main characteristics of an amplifier, these are:

- Noise.
- Distortion.
- Bandwidth.

Noise is related to the type of elements employed on the feedback network and the model of the selected active device to create this first stage. By means of an already known equations it is possible to calculate the equivalent noise found at the input port of the amplifier according to the topology and amplifier type. About the distortion design constraint, this relates to the kind of distortion known as *clipping distortion*; this distortion occurs when the output signal level is higher than the maximum operation level of the output amplifier device. As a consequence, the output signal can be seen "clipped" by certain amount, at this stage the objective is to keep the signal within a certain acceptable range. Last, the third stage to be designed relates the bandwidth capabilities of the overall amplifier. In order to obtain an estimate of the maximum operational frequency, by means of a procedure known as *LP Product* it is possible to calculate the maximum bandwidth that the amplifier could handle.

It is important that the developed tool should be easy to operate by any user no matter the experience on circuit design matters. Therefore, by means of a graphical user interface it is possible to reduce the human interaction as much as possible. By the use of icons, buttons and on-screen instructions it is possible to guide, in a simple way, the designer through all the design stages.

Once the design process has been concluded, it is necessary to provide some kind of useful output to the user and the means to perform the verification of the design. When the designer has reached the end of the process, a summary is displayed showing the desired specs, the passive and active devices selected to achieve the desired goal. On the other hand, the verification should be carried out employing verification electronic tools like Spice or any other circuit simulation software , this way the design is evaluated and the operational characteristics are validated.

Finally, three examples are provided to verify the tool capabilities. First, a trans-impedance circuit is shown because it has just one feedback element in a single loop topology. Second, the complexity is taken one step forward with a single-loop voltage amplifier. This circuit has two feedback elements within the loopback, the design should be developed taking into account the contributions added by these elements. Third, the most complex topology that this tool can handle is the two-loop topology, therefore makes it ideal to show the strength of this CAD tool.

Resumen

El diseño tradicional de circuitos utiliza como punto de partida un circuito con características similares a las que se desea cumplir. Una vez seleccionado el circuito, se modifica de tal manera que cumpla las condiciones de operación para el nuevo diseño. Desafortunadamente esta manera de diseñar circuitos electrónicos tiene varios inconvenientes, los dos mas importantes son:

- El tiempo que se invierte en adaptar el circuito a las nuevas condiciones de operación es muy alto.
- En caso de que no exista un diseño que se pueda adaptar, es necesario invertir tiempo y recursos para crear uno nuevo.

Este trabajo de tesis presenta el desarrollo e implementación de una herramienta para automatizar el proceso de diseño de amplificadores con retroalimentación negativa basado en la teoría de *Diseño Estructurado de Circuitos*. El fundamento principal de esta teoría es establecer una solución ideal y, a partir de esta, se procede a realizar las modificaciones necesarias para poder llegar a la solución particular del problema. El elemento base que utiliza esta metodología es el *nullor*, el cual es un elemento ideal de dos puertos con las siguientes características principales:

- En el puerto de entrada el voltaje y la corriente tienen valor de cero.
- El puerto de salida tiene valores de voltaje y corriente que siempre cumplen los requerimientos de la carga.

En este trabajo, los amplificadores a realizar serán aquellos con con uno y dos lazos de retroalimentación. Las topologías que se pueden implementar son: amplificador de voltaje, amplificador de transconductancia, amplificador de transimpedancia y amplificador de corriente.

Debido a que el nullor es un elemento ideal, es necesario realizar la síntesis de este dispositivo utilizando elementos "reales" tales como resistencias, capacitores, transistores, etc. La síntesis se lleva a cabo por medio de dispositivos activos, ya que son los únicos que pueden, hasta cierto punto, recrear las características de amplificación que posee el nullor. Sin embargo, los dispositivos activos tienen ciertas limitantes como lo son: generación de ruido, frecuencia de operación limitada o un rango estrecho de valores de alimentación. Los dispositivos activos en este trabajo se han modelado utilizando sus *modelos de pequeña senñal*, es decir, el dispositivo activo se representa utilizando elementos capacitivos, resistivos

y fuentes controladas. La razón para emplear el modelo de pequeña senãl radica en que este trabajo busca obtener características que forman parte del comportamiento en AC del circuito mientras que solamente algunos valores de DC serán tomados en cuenta solamente en ciertas partes del diseño.

La metodología de diseño divide el problema en tres problemas mas pequeños y que representan las características mas importantes de un amplificador, estas son:

- Ruido.
- Distorsión.
- Ancho de Banda.

El ruido está relacionado con el tipo de elementos utilizados en la red de retroalimentación y con el tipo de dispositivo activo seleccionado para implementar esta primera etapa. Se cuentan con las ecuaciones para obtener el ruido equivalente a la entrada del amplificador de acuerdo a la topología y el tipo de amplificador. El tipo de distorsión que se quiere evitar es la llamada *distorsión por clipping*; esta distorsión llega a ocurrir cuando el nivel de la señal de salida es superior a los niveles máximos de operación del dispositivo amplificador. Como consecuencia se puede observar que la señal a la salida esta "recortada" una cierta cantidad, en esta etapa el objetivo es mantener esta señal dentro de límites aceptables. Por último, la tercera etapa se refiere al ancho de banda que puede manejar el amplificador por completo. Para obtener la estimación de la frecuencia máxima de operación, se aplica un procedimiento llamado *Producto LP* el cual provee de una estimación de la capacidad en frecuencia del amplificador de una manera rápida y precisa.

Con el fin de llevar el proceso de diseño lo más simple posible, se ha pensado en utilizar una interfaz gráfica. Esta interfaz, haciendo uso de íconos, botones y texto informativo, permite al usuario realizar el proceso de diseño con la menor intervención posible.

Una vez concluído el proceso de diseño se debe de proveer al usuario de una retroalimentación gráfica y, además, generar información para que pueda ser utilizada en la la verificación del diseño terminado. Pensando en lo anterior, al final de la operación de la herramienta se muestra un resumen con los datos de las especificaciones deseadas junto con los datos de los elementos tanto activos como pasivos que cumplen dichos requerimientos. Por otra parte, la verificación debe realizarse utilizando herramientas tales como Spice o alguna herramienta de verificación de circuitos para obtener de forma cuantitativa las características de operación del amplificador por completo.

Finalmente, a fin de demostrar la utilidad de esta herramienta se muestran tres ejemplos de diseño de amplificadores. El primero de ellos es un amplificador de transimpedancia el cual cuenta con solo un elemento de retroalimentación dentro de un lazo simple. Para el segundo ejemplo se muestra un amplificador de voltaje, este caso es mas complejo ya que ahora son dos elementos resistivos en un solo lazo de retroalimentación. Para concluír, el tercer ejemplo muestra el diseño de un amplificador de voltaje con dos lazos de retroalimentación.

This work is dedicated to My Wife and Daughter. Their continue support through the tough times has been nourishing. Sorry for the inconveniences.

To My Father and Mother. Their endless love and support through all these years has to flourish. I guess the time has come.

To my Advisor L. Arturo Sarmiento Reyes Ph.D.Sc.. Thank you for your support, advices and ultimatums. I guess that I would have ended in nowhere wihout your thoughtful pointers.

To Luis Hernández Martínez Ph.D.Sc.. You were the first to believe that I would fit in the CAD group. My deepest steem.

To my very good friends at the INAOE. The administrative personnel and the people at the Electronics Department. My best regards.

To CONACYT for granted me a scholarship and supply upkeep, without it I would never be able to accomplish this goal.

Contents

List of Figures

List of Tables

Chapter 1

CAD Tools for Electronic Design

1.1 Introduction

The production of any electronic circuit is concerned with successively taking a requirement; developing it; producing an accurate abstract representation (a circuit diagram); making, evaluating, and testing it; and then returning to the original circuit design to correct errors. This process is iterative, frequently extending across the lifetime of the circuit, with modifications and improvements introduced and specifications changed continuously. This way the development process involves the construction and manipulation of various (often very large) representations of a circuit and the examination and testing of these with respect to the initial requirement. The development of any sub-circuit representation for running a test can be long and complicated. Performing a complete set of these design activities for a system of any size is long and, specially, costly.

The computer's ability to contain large amounts of data and to facilitate its access and updating in a dynamic way, besides of course with powerful programs for aiding in the synthesis and analysis of circuits, is a major for their use.

With each new level of refinement, more information concerned with the detailed physical and functional implementation of their circuit is included in the description. The design process involves transformation between these representations during both synthesis and verification.

There are many programs written designed to produce one representation of the design process and its results given to another tool. However for many of these tasks, programs are not perfect and much human intervention is still needed. For large designs, this is very expensive because more than one product iteration is frequently required to remove all errors before a working design is produced. The cost of such iteration is often of the order of some $1000000 [2].

The term CAD *framework* has come to mean all of the underlying facilities provided to the CAD tool *developer*, the CAD system *integrator*, and the *end user* (IC or system designer) which are necessary to facilitate their tasks.

Broadly seaking, these three groups of people represent the user of the CAD framework, each with their own needs and particular emphasis. The CAD framework plays an analogous role in the development of engineering-specific, or even electrical-engineering-specific, software systems to the role played by an operating system for the development of general-

purpose software applications, or the role of a specific programming environment for software development in a particular programming language [3]

Despite the fact that many tools can perform many tasks that help the development of new technologies, there is an area that needs effective tools in order to automate a process that is cumbersome. The area is related with the *synthesis* process. This work is aimed to develop a CAD tool to automate the synthesis process, starting from the synthesis at the highest level (architectural) down to the circuit synthesis.

1.2 CAD Framework

The traditional "waterfall" model for software development involves a cycle of specifications and reviews between developer and client. This approach makes sense under the following conditions [3]:

- The client knows precisely what is required.
- The product is not needed in some time.
- The product can not be acquired by any other means.
- The client is a unified entity.
- The requirements will not change significantly during the life of the project. Unfortunately, these conditions do not fit well with CAD framework development

The conditions under which the Cooperative Development Method (CODEM) [4] succeeds are quite different:

- There are many potential clients with similar, though not necessarily identical needs.
- Some software exists which solves part of the problem and which can be used as a common starting point.
- An organization exists which can serve as a focus, both taking responsibility for managing communication between the co-operating parties and for integrating and distributing the emerging software system.
- The software selected as the starting point is modular, with well defined inter-module interfaces.
- The selected software is available to all interested parties.

The CODEM approach replaces the typical "waterfall" model for software development with a loop involving three steps:

1. Build a working prototype.
2. Determine the most significant weakness in the prototype.
3. Develop a solution and return to Step 1.

1.3 Commercial CAD

Some organizations rather than developing their own tools for process, device, circuit or any other process involved on design and development process spend their resources on acquiring commercial CAD frameworks. This tools can be comprised of several (and often optional) modules or in a single package there is a complete suite of options for simulating electronic and electric behaviors.

For this kind of tools there is a trade off on which the development is reduced but the investment has a high cost because commercial tools are offered with time expiring licenses (usually a year or two). After this license expires it is necessary to renew it or the software is unable to perform.

1.3.1 Synthesis

With the explosion in communication and consumer products, analog design is looming large. However, analog designers - and tools to enhance their productivity - are in very short supply. Development of the analog segments of sophisticated designs may take longer than the digital portion, even though the analog portion is often smaller. Meanwhile, digital designers have synthesis, formal verification, and test pattern-generation tools, among others, at their disposal. They also can create large circuits from high-level languages.

So how can greater productivity be achieved in analog design? The highly anticipated solution will let designers synthesize analog and mixed-signal circuits. But until this capability is developed, designers must rely upon evolving answers to the analog challenge.

The downside to using top-down approaches in the analog realm is that they require a paradigm shift in the way that analog circuits are typically designed. Before top-down analog tools gain wide acceptance, their users must expand their traditional schematic-based orientation to include the language-based orientation necessary to take full advantage of such tools. Unfortunately, most engineering schools aren't yet teaching this language orientation.

Setting aside these considerations of approach and orientation, it seems clear that the effective use of specialized software can help to address the shortage of qualified analog designers. Conventional wisdom holds that it requires about an order of magnitude more time and effort to create and document a design with reuse as a goal than it does to produce the same design for one-time use. A handful of tool vendors are working to bring synthesis to fruition by developing tools that address one or more of the three major phases of synthesis: topology creation, design optimization, and design layout. With such tools in early development at this time, their promise is still largely latent.

Recently, electronic and software developers and researchers have been involved in bring automation to the synthesis process [5, 6, 7, 8, 9]. This automation is based mainly on VHDL-AMS which is an analog description language derived from the VHDL description language employed for digital synthesis. Unfortunately VHDL-AMS is not an standard for the industry since some companies develop their own analog hardware description language. Until the industry vendors unify their efforts towards an standard there is not a hope for a fast synthesis tool in the near future.

1.3.2 Verification

This section has the most developed tools. Most of these tools are employed for simulation (circuit, device, process). Circuit simulation tools can be divided into:

- Electrical Simulation Tools.
- Electronic Design Automation Tools (*EDA*).
- Specific Task CAD Tools.

1.3.2.1 Electrical Simulation Tools

This kind of tool performs simulations based on a netlist. Netlist is the way a user provides circuit description, type of analysis to perform and the way results are going to be saved and displayed. Basically is a description language for circuit analysis. The tool can perform analysis such as: DC, AC or Transient. Complex tools can do other numerical analysis like Monte-Carlo, noise or even *S*-parameter calculation.

Here is a list with some programs that falls into the simulation tools category:

- Spice [10] - Is by far the most known tool for circuit simulation. It has been used on the electronic development since the early 70's and has been adopted as the basic tool to verify if the design accomplishes design specs that the designer expects. Actually this simulator is not developed, however its source code is provided for further optimization free of charge.
- ngSpice [11] - Based of the Spice 3f5 code for device simulation, this tool is aimed to provide a mixed-mode/mixed level circuit simulator. It can accept netlists generated for Spice. This project is under constant development and attempts to be be up to date.
- PSpice [12, 13] - PSpice is a graphic front end for analog and digital circuit simulation. It is based on Spice and is currently included in the OrCAD design suite.
- HSpice [14] - Originally release by Avant! now it is owned by Synopsys. This software is employed by industry and research institutions to perform analog and digital simulations. Based on the Spice code it provides better numerical algorithms for the DC, AC, transient, Monte-Carlo analysis. Many device manufacturers provides device models to be employed on this simulator. In order to operate this tool a license must be purchased.
- gNUCAP [15] - Similar to the ngspice circuit simulator this software package is programmed without being based on the Spice source code. However, some models are based on the Berkeley models. Its main goal is to provide a true mixed-mode simulator. This tool is free.
- APLAC [16] - This tool was developed aimed to the cell phone industry of Findland. It is funded by Nokia mobile phones. The software is mainly employed for simulating RF systems. It has a simulator engine based on Spice and also is able to simulate latest

technologies like MEMS (Micro-Electro-Mechanical Systems) and Bluetooth. This CAD software has two license options; the first is a commercial license which has to be renewed after some time and the second one is a student license making it capable to use all the features but with memory restrictions.

1.3.2.2 EDA Tools

These tools are used for electrical circuit design, schematic capture, simulation, prototyping, and production. These tools offers a suite of applications for electronics design, including schematic capture, attribute management, bill of materials (*BOM*) generation, netlisting, analog and digital simulation, and printed circuit board (*PCB*) layout.

Often complex to use and expensive, EDA tools are aimed to simplify design process by providing the most complete set of tools that a designer could use. Figure 1.1 shows an EDA environment.

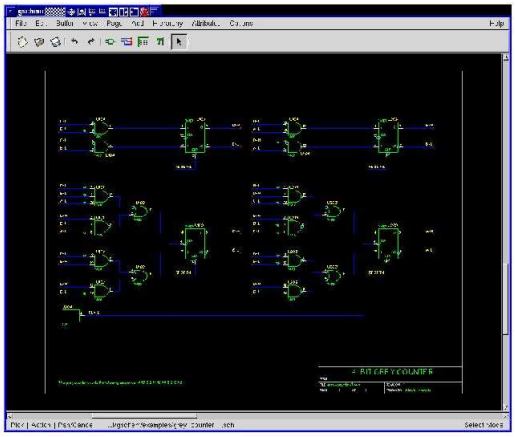

Figure 1.1: EDA application screenshot

Here is a list with the most common EDA tools:

- Cadence [17] - This CAD tool is perhaps the most employed design software on the industry. It can be customized to satisfy any design requirement by offering add-on packages. Since it is one of the referrals on the industry its price is high even with the basic options. Some of the most used packages for analog design are:
  - Virtuoso Analog Design Environment - Is the analog design and simulation environment for the Virtuoso custom design platform.
  - Virtuoso Spectre Circuit Simulator - Provides fast, accurate simulations for tough analog and mixed-signal circuits.
  - Virtuoso Schematic Editor - The design composition environment for the Virtuoso custom design platform.
  - Dracula - This is a verification CAD tool and is independent of the design methodology - bottom-up, custom, standard-cell, structured gate array, or block-oriented.

- OrCAD [18] - This complete EDA design suite developed by Cadence is comprised of several tools, among them are:

  - OrCAD Capture - Schematic entry system makes possible to create a schematic by selecting and placing devices in a graphical way. Once performed the schematic the software can translate it into a netlist for simulation using PSpice, Spice or other type of netlist. It can even provide a file to produce the PCB.
  - SPECCTRA for OrCAD - This program was used within the PCB editor and now is offered as a separate module, this powerful software is capable to perform autorouting for a PCB project.
  - OrCAD PCB Editor - Is capable to produce custom layouts or special PCB projects like ISA, PCI, cards.

  OrCAD is an expensive design suite. It can be purchased with only the basic options and add more as needed.

- gEDA [19] - This is a set of various CAD tools grouped in order to provide electrical circuit design, schematic capture, simulation, prototyping and production. This software is provided for free and is possible to install only the needed tools without the need to install the complete suite. It is provided for free.

1.3.2.3 Specific Task CAD Tools

The previous CAD packages are able to perform a broad range of tasks that involve the design process. By taking into account the customer's specifications, the designer by using one or more of these programs can achieve and suffice those requirements and deliver a final product which satisfy the customer's needs.

It is possible to develop a program to satisfy a particular need. Not only could it be written in a particular programming language such as C, FORTRAN, C++, PASCAL, it may be done by using non-CAD programs like Maple, Mathematica or Matlab. The program, possibly, is not so attractive on visual design but one thing is certain, it achieves that specific result that some other, and expensive tool, is not able to produce. On the other hand, if the designer needs a tool to automate a process to speed up the design time or to develop a new method to achieve the electronic design, the answer is to develop a CAD tool for a specific task.

Commercial and non-commercial applications have been developed to satisfy a particular need to achieve a goal. These tools can be found as add-ons for programs like Maple [20], Mathematica [21] or Matlab [22]. Also they can be found as a library for a programming language or as a standalone application. Some examples of this kind of applications are:

- Analog Insydes [23] - This is an add-on package for the Mathematica [21] software. It is capable to perform DC, AC and transient analysis. Provides graphic output of plots like Bode or root-locus. It is capable to present the symbolic equations of the system. Despite the fact that this software relies on libraries provided by Mathematica, is a high cost CAD tool.

- Syrup [24] - Another example of a CAD tool based on a math CAD package. This Maple based software, developed by Joe Riel, solves the MNA of a electric circuit symbolically. It accepts a Spice-like netlist input but modified to be processed by Maple. Provides the same kind of devices as Spice. This package is given for free.

- AMSCiDe [25] - This application is a high-level language that supports the IEEE 1076.1-1999 Standard VHDL Analog and Mixed-Signal Extensions, known as VHDL-AMS. The application has included mechanisms that allow modeling and faults simulation. In follows, the CAD tool will make referenced as AMSCiDe (VHDL-AMS Circuit Design). This tool represents a first approach of an integral design environment that can be used in diagnosis processes and mixed circuits verification.

For analog design, there is a lack of tools capable to take the design process from specifications to the complete netlist of devices that fulfills these specs, without human interaction. The need to develop one has been arisen as the time to develop and test have become critical nowadays. This not mean that an AHDL must be developed and integrated into a tool; the approach must be simple and easy.

1.4 Design Methodology

This point refers to the guidelines that the developer could or should follow to complete the task of circuit design and fulfill a need. In like manner to analog synthesis, the design methodology does not have the solid foundations to provide one or two methodologies that guarantee a successful design. The most used methodology is to resort to previous, already proven, designs and find the one that is similar to the need. Consequently this design is modified to achieve a desired performance. In case that there is not an adequated design the design process must be done from ground up.

As seen the trial and error process is long and tedious, not to mention the huge amount of time invested on development. There is a need to research, develop and establish methodologies not bounded by special needs but to provide a general knowledge for analog circuit development. This methodology must be accurate and fast.

Here are some methodology proposals:

- Genetic Algorithms [26, 27]
- Macromodeling [28]
- Top-down approach [29]
- Hierarchical design [30]
- Structured Design [31, 32, 33]

The most general methodology until today is Structured Design which embrace the Top-Down approach and the Hierarchical Design. Yet powerful the Structured Design is not widely employed but it has made its way to be ahead other methodologies for analog design.

In conclusion, there are some design methodologies capable to ease the process but still there is not one capable to be translated into a commercial CAD tool. A commercial CAD tool automated in such a way that the designer (user) inputs some values concerning the parameters that the design should comply and by a click the program performs the required calculations providing an output showing the necessary devices, bias values and placement. Maybe in a near future this kind of tools could be found in laboratories, industries and classrooms but as electronics evolves the need to keep an up to date software package means time and money investment that not so many people can afford.

1.5 Object Languages

All computer languages are essentially ways of representing and modelling some aspect of the real, or imaginary, 'world' on a computer. Among the problems concerning both the computer scientist and the practitioner of artificial intelligence, one which undoubtedly prevails is that capturing in this formal way, a sufficiently large number of problem's aspects from which to build a complete model.

It is interesting to note that both the computer scientists and those engaged in knowledge based systems which, in the early days of their subjects were following different paths, have, in their approaches to this problem begun to show similarities - namely the object in computer science and the notion of the frame in artificial intelligence.

1.5.1 Traditional Problems

In imperative procedure oriented languages - like PASCAL, C, C++ or Java - it is more or less established that program and data be kept separate, and furthermore, that the former be active and the latter passive. The ability, when using such languages, to decompose the problem as clearly as possible into these two entities is central to success.

Problems with side effects are sometimes encountered in the use of such programs, usually stemming from two or more procedures each independently trying to assess the same piece of data. Functional programming is one of the forms of development that has attempted to tackle this problem.

1.5.2 Differences Between Object and Conventional Representation

In the object oriented technique, programs are not partitioned into separate and distinctly identifiable procedures and data. Instead, the basic program construct is represented by a data type called an object which has the attributes of both procedures and data. All information relating to the representation and behavior of that object is contained within the boundaries of that construct.

For example, in a CAD system, one may wish to perceive the same component in more than one manner - as a screen entity with x and y coordinates, as a value of capacitance, as a physical device with size and so on. Procedures such as those required to move the device's

picture around a screen may be incorporated within these descriptors. All this information would be contained within a single unit referred to by one label [2].

1.5.3 Classes and Hierarchies

Another important feature of this paradigm is the fact that one may create a single generic description of object 'class' which is an abstract and formal description of a particular artifact or concept. Class instances, or realizations of individual artifacts, are created simply by requesting that a new object be added to the current environment and by passing distinguishing actual parameter values to the new entity. In practice, objects may be arranged in a network which reflects the structure of the modelled system. For example: a three level hierarchy of objects may be used to express a commonly used circuit.

Some object orientated systems support an ability to instantiate a whole network with a single command. A simple request to instantiate a block to be synthesized say, would automatically bring into existence a complete circuit comprising methods describing in detail its behavioral, physical, and structural aspects. By being able to make an indirect reference to a tightly associated group of components a far greater clarity of expression and confidence of correctness is attained than is generally the case following the separate calculation of all parameters.

1.5.4 Programming Languages

To have a way to implement an algorithm is important but the most is to code it in a language capable to perform complex calculations and capable to be embedded into a graphical interface. Java is one of the latest programming languages focused on the easy way to code, capable to provide a nice graphical interface and the main feature is that it is operative system independent. Although this language is not optimized for complex calculations and long mathematical routines.

C++ is an object oriented language capable to perform complex mathematical routines, complex calculations and graphical interfaces can now be programmed easily. It has native libraries for almost any task. On the lower side it is platform dependent, that is, it needs certain resources tightly linked to the operating system it is developed on.

CAD tools are almost completely developed in C/C++ and runs on the most popular operating systems as Windows©, Linux© and Solaris©. The most powerful tools runs on Linux or Solaris. Developing on these platforms is easy since there are development tools to speed up the process and also share compiler packages so porting from one platform to other is not a complex task.

1.6 Thesis Objective

This work is focused on developing a CAD tool to automate the amplifier design based on principles given by Structured Design. Main goals are:

- Implement a CAD framework.

- Translate into a program language design algorithms based on Structured Design.
- Design and program a tool capable to automate analog amplifier design.
- Offer a graphical user interface (*GUI*).
- Provide an output to be simulated on electrical simulation programs.

1.7 Conclusions

Analog design is the foundation for up-to-date electronics and designs. Development of this kind of technology will continue as long as electronics evolves. On the CAD market there are many tools that helps designer to keep the research and development process fast and accurate.

It has been shown that the principal manufacturers of electronic devices like IBM, Intel or AT&T spend time and money resources in developing CAD tools to fulfill their need for specialized software. Despite the fact that many commercial tools provide GUI interfaces to perform schematic capture, simulation or PCB design still there is not a CAD package capable to automate entirely the synthesis process. The analog synthesis process starts with constraints provided by the designer and a software tool, which is based on a design methodology, performs circuit synthesis from the architectural level (highest) down to the device level (lowest). Process ends when results are provided by the tool, and they can be verified by an external circuit simulation tool.

Finally the thesis objective has been presented which is to develop a tool being capable to fulfill the designer's requirements in a fast and accurate manner.

Chapter 2

Design for a Specific CAD Tool

2.1 Introduction

The main goal to achieve on this work is to produce an electronic CAD framework aimed on the design of amplifiers, specially the synthesis of the design is a process that consumes large amounts of design time.

On Figure 2.1 illustrates the transformations involved on the design process of an amplifier.

It is necessary to establish an order to base the design process and hence the foundation for a solid design methodology. First is to structure the objects in an order, this should act as a raw model for the overall strategy. For example, it is not possible to perform synthesis on a design which is incomplete. The design process should have this order:

1. Design Methodology
2. Synthesis

Additionally to the strategy for design and program a CAD tool, another important issue to address is the way the tool interacts with the user. In computer science terms, the user interface (UI) that lets to introduce basic data, retrieve information from it, save it, display the results of calculations or inform the user that an abnormal situation have occurred.

2.2 Design Methodology

Design methodology for analog circuits, refers the theory and steps involved in order to develop a circuit. As explained in Chapter 1, there are various methodologies capable to tackle the problem of analog design. Nevertheless, not all are capable to not only provide a design path but to ease the synthesis of the circuit. In addition to the design and synthesis problems another concern should keep the attention to the developer and is the feasibility to translate the selected methodology into a computer program.

In order to develop a software CAD tool the design methodology and synthesis methodology ought to be clearly defined and structured in such a way that program coding is kept clear and simple. This relates whether is an VHDL-AMS compiler-interpreter, simulation tool or a special purpose tool.

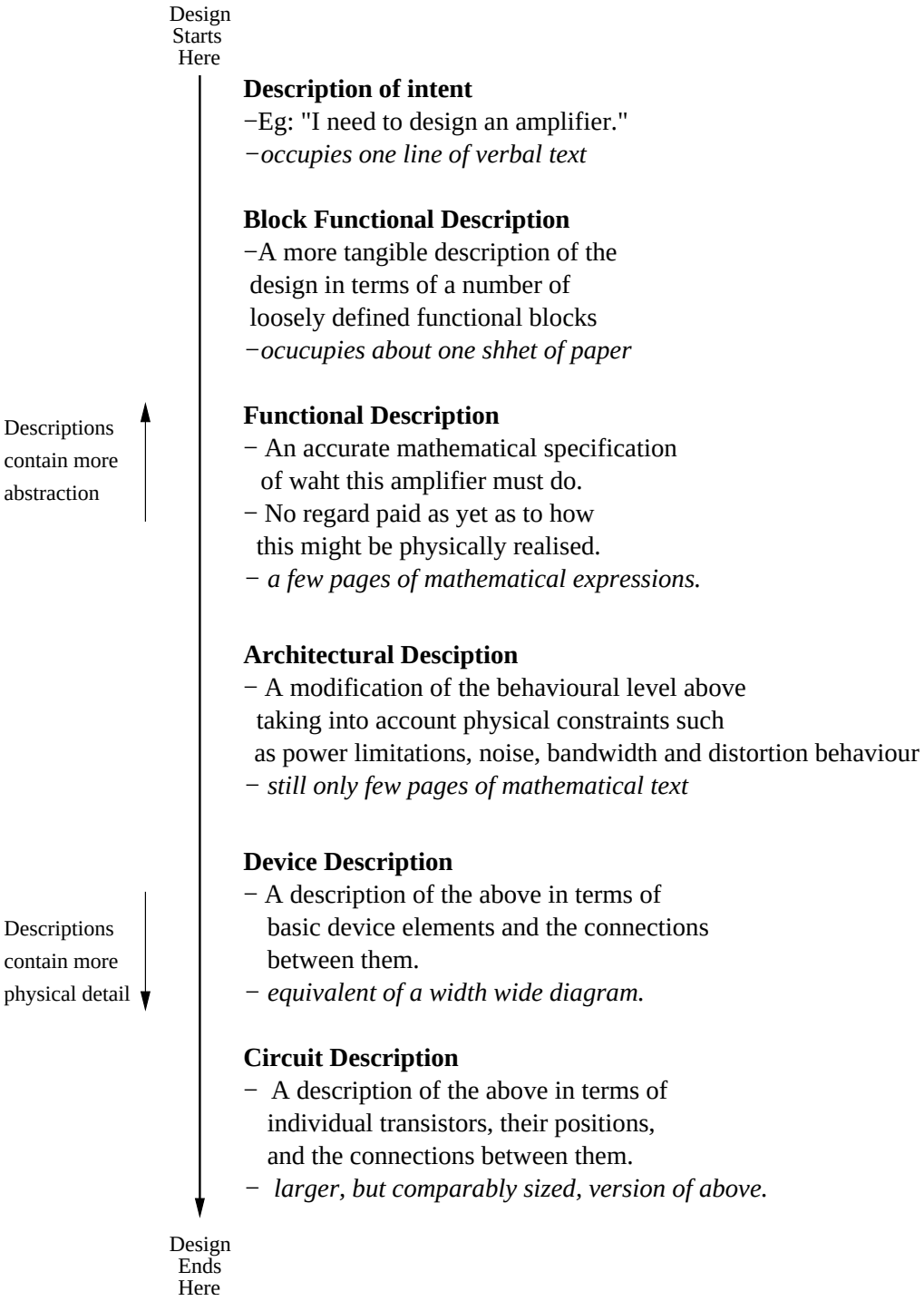

Figure 2.1: Amplifier design as a series of transformations.

2.2.1 Structured Design

In the 80's Ernst Nordholt on his Ph.D. thesis [34] proposed and presented a new way to do analog design. He showed that it is possible to divide the whole design into smaller blocks and these blocks developed without involving other blocks, that is, to build one block at the time. First considered simple models and as design evolved more complex models were involved for certain blocks. In other words, he structured analog design in such a way that it made possible to establish some basic steps to be followed and obtain a new design. Since those early days, works like Verhoeven [1] and Stoffels [33] have taken these basic steps and taken further.

Nowadays structured design is another alternative for analog design to develop new designs in areas like amplifier design [34], [33], [35], [32], [36], bandgap references [37], [38], oscillators [39], [31], [40], [41], and it can even be applied to filter design [42], [43], [44], [45].

In order to understand the way structured design works it is necessary to explain the basic aspects that this approach is based on. The main goal for this kind of design is to obtain as fast as possible circuit synthesis without losing accuracy for the final design.

Shannon [46] provided a formula in order to express the performance of a transmission line, it can be also applied to specify the performance of electronic circuits. This formula shows that only three aspects can affect performance for a circuit:

- Noise (N)
- Signal Power (S)
- Bandwidth (B_w)

related to each other via this formula:

$$C = B_w^2 log\frac{S+N}{N} \tag{2.1}$$

where C is the signal-handling capacity of the circuit. It is a measure for the information that the circuit can handle per second. It can be seen from (2.1) that the bandwidth is linear and the signal to noise ratio is in the algorithm, so an increase of the bandwidth yields more improvement than an increase in the signal to noise ratio. However, under low-power conditions or at very high frequencies the signal handling capacity has to be optimized via the signal to noise ratio.

In practice noise, signal power and bandwidth are not only to describe circuit performance; there are other aspects such as supply voltage, supply current, power consumption and so on. These extra specifications can be seen as parameters within the three basic aspects.

2.2.1.1 Orthogonality

As explained before there are three fundamental aspects that have to be provided and then optimized. During circuit synthesis an orthogonal approach is applied for the synthesis, it is done with respect to the three fundamental aspects. This way turns a multi-dimensional

design problem (noise, distortion, bandwidth, supply voltage, supply current, power consumption and many others) into a three separate one-design problems easier to resolve.

It is obvious that in practice these aspects are not completely orthogonal, nevertheless during design process *it will be assumed they are orthogonal*. The design strategy applied to develop the CAD tool is to make noise, clipping distortion and bandwidth orthogonal. For the basic three design aspects these assumptions on orthogonality hold:

- Noise
  When noise is evaluated clipping distortion is not considered. Bandwidth for the complete circuit is not taken into account. Frequency behavior is considered when noise performance is evaluated and small-signal models for the components can be applied.

- Clipping Distortion
  When clipping distortion is designed and evaluated neither noise nor frequency are considered. Static large-signal models are used. Noise produced by this stage is assumed to be small enough to obtain a negligible correlation with the non-linear behavior of the circuit, nevertheless it is necessary to perform another noise evaluation just to make sure the overall noise performance is within constraints.

- Bandwidth
  Bandwidth evaluation is performed without taking into account clipping distortion and noise behavior. Small-signal models are used for this stage.

2.2.1.2 Model Simplification

To make a fast synthesis it is necessary to identify feasible solutions in a simple way. To avoid long and time consuming calculations needed to identify these non-feasible solutions, certain criteria must be formulated aimed to obtain a desired performance. In case the criteria is not met it is a fact that the final solution never meet its specifications. When criteria meets there is a *chance* the specifications will be met in the end; by this way it is reduced to the minimum the risk that a solution is non-feasible.

Fast evaluation criteria is obtained by defining simple models. Evaluation is eased to a great extend and it assures actual solution shall never perform better than the ideal solution. If ideal solution does not meet specifications so the actual solutions, though. Models must be arranged in the design in such a way that actual performances can come very close to the ideal bahavior.

2.2.1.3 Hierarchy

Different complexity levels can be distinguished throughout design process. Starting from simple models design is taken into a refining process producing higher accuracy but on the other hand calculation complexity is increased as well. On every level decisions must be taken. Efficient design strategy is the one that on every decision taken it is assumed correct and unaltered for the rest of design. Orthogonality is useful for hierarchy in the design process because makes possible to take a decision for one fundamental criteria without taking the others into account.

Hierarchy is a good means to reduce complexity of the design process. For instance, in a negative-feedback amplifier two blocks can be distinguished: the feedback network and the active part. By applying hierarchy it is possible to design each block assuming ideal behavior of the other. Then, instead of design a complete negative-feedback amplifier it is divided into a more simple design of two smaller sub-circuits. For these two sub-circuits further subdivisions are possible. This way it is possible to identify and point out possible bottlenecks and establish strategies to avoid them. Certainly the orthogonality of the three fundamental aspects noise, power and bandwidth is of great help to establish hierarchy of the design.

2.3 Synthesis

Once the design methodology is chosen, the next step is to proceed and establish the way that the circuit is going to be build. This refers to how this design evolves from the general structure provided by the methodology required by the design, which is the highest level for synthesis, down to physcal devices, currents, voltages and topologies.

This work focuses on the design of negative feedback amplifiers and the automation for this process. First the structure of negative feedback amplifiers is explained. Later on two basic concepts are introduced. The first relates to the idea provided by structured design to start the design process taking the ideal solution. This ideal solution is synthesized into the nullor, an ideal element. Secondly a very useful concept is presented, the superposition model. Subsequently, by applying the nullor concept and the superposition model, basic configurations for one and two loops of feedback are shown. The last part of this section is devoted to explain how the noise, bandwidth and distortion can be synthesized.

2.3.1 Negative Feedback Amplifiers

An amplifier is a circuit that increases the power contents of an information carrying signal by multiplying it by an accurate constant shown in Figure 2.2. The Chain-matrix of an amplifier has four entries that must be constant and accurate. Evaluating Chain-matrices of various devices and ranking them two different class of devices can be defined:

1. Devices with an accurate and constant transfer.
2. The others.

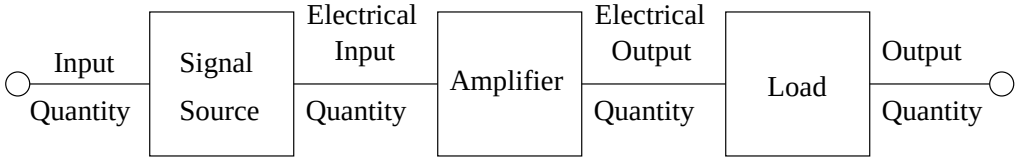

Figure 2.2: Amplifier scheme.

The first class contains only passive devices, which are not capable to produce any power amplification, on the contrary, reduce power from the original source. Second class of devices

can provide power amplification but none of them has sufficient accuracy. It is necessary to develop a strategy capable to combine the best of both classes. Circuit topologies have found useful to use the accuracy of devices from the first class and the power amplification for the second class. The only transfer that can be obtained from the devices of the first class is a transfer with reduced power of the information it transfers. Only is possible to make very accurate attenuators. Feedback topology is the only amplifier topology that makes use of an accurate attenuator. With a negative feedback topology is possible to design a perfect amplifier.

On this work the operational amplifier is not used because this device is intended for use in a wide range of applications, and as a consequence, design for specific purposes suffer from suboptimal performance if based on a generally applicable operational amplifier.

The design is based on the fact that the function of amplifier is to enhance the energy level of information provided by a signal source and minimal affecting the amount of information present in the input signal. Departing from a complete specification of the signal source, the amplifier load and the desired transfer of the input quantity, an amplifier is synthesized optimizing the behavior of the amplifier for various quantity aspects.

On Figure 2.3 the nullator indicates that the voltage difference between the two nodes it connects are defined to be equal and by definition no current flows through the nullator. So the nullator itself has no ability to force this equality. This is done by a controlled source U_{src}. The value of this source is controlled in such a way that the conditions imposed on the circuit by the nullator are met. This variable source that adapts its output to fulfill nullator conditions is called a *norator*.

With voltage source U_{src} acting as norator at the input of the voltage divider and the nullator at the output of the divider accurate gain is achieved, entirely dominated by the value of the passive components. The voltage across the norator is given by:

$$U_{src} = \frac{R_1 + R_2}{R_2} U_{in} \tag{2.2}$$

It can be concluded that is possible to build amplifiers of which the transfer is completely dominated by accurate passive components using the negative feedback topology. The only requirement for this is the availability of a nullator and a norator.

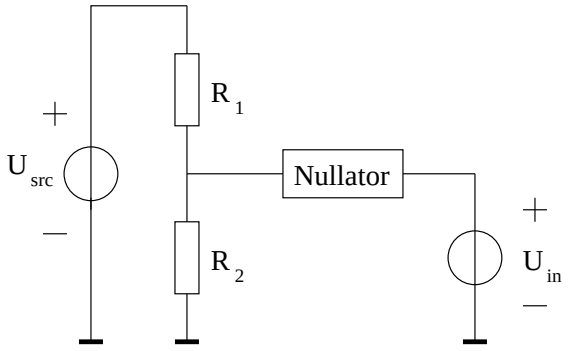

Figure 2.3: Voltage divider with a nullor.

2.3.1.1 The Nullor

Since it is common that nullators and norators occur in pairs in circuits it is convenient to define a network element that represents this combination. This element is known as *nullor*. It is a two-port shown in Figure 2.4. Its Chain-matrix that relates input and output quantities are:

$$\begin{pmatrix} u_i \\ i_i \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u_o \\ i_o \end{pmatrix} \tag{2.3}$$

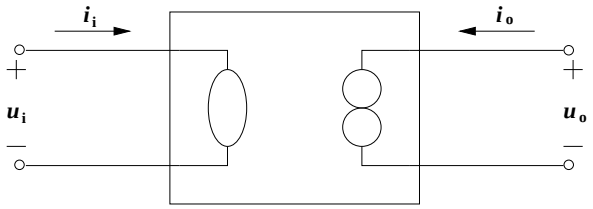

Figure 2.4: The nullor.

As seen the Chain-matrix of a nullor contains only zeroes. A nullor is an ideal element with infinite voltage gain (μ), current gain (α), transconductance (γ) and transimpedance (ξ). It adapts its output such that at its input terminals the voltage and current are equal to zero.

It is possible to implement nullors with active device components and consequently implement amplifiers that have the accuracy and constancy of the passive devices for the first class and use the gain devices in the second. The negative feedback topology makes possible to split the design into two orthogonal parts:

1. The design of the feedback network, assuming a nullor is present.

2. The implementation of the nullor with suitable active devices.

From the first step the fundamental limit to the performance can be found. The nullor produces no noise, distortion and does not limit the bandwidth of the circuit in any way. It is perfect, as shown before. Practical nullor implementation never improve performance found with the nullor. In case that there is an improvement the circuit modelling is incorrect or insufficient, therefore most be corrected.

After first step is completed on success and performance is within specifications, the active circuit that fulfills the nullor is implemented. When a resulting circuit is not performing according to the specifications, it is just a matter of finding a better nullor implementation to modify circuit behavior within specifications.

2.3.1.2 Superposition Model

Amplifier design consist in two major parts as it has been shown on previous sections. Feedback network design and the design of the nullor, an scheme is shown in Figure 2.5. The asymptotic gain model is used to perform these two design steps independently, Figure

2.6. β is the transfer of the feedback network, A is the transfer of the active circuit (nullor). Frequently a direct transfer exists between input and output caused by various kinds of parasitic coupling, which is been modelled via the direct transfer factor A_{t0}, i.e. the gain when the loop gain is zero. When the input of the active part is not connected directly to the source and the output of the feedback network, the extra transfer is modelled via ξ and in a similar way ν models an extra transfer between the norator and the amplifier output when they are not directly connected. A_{t0} generally is a property of the active circuit itself. In a "proper design" ξ and ν should both equal unity, though frequency compensation components at the input or the output may affect these parameters in this case.

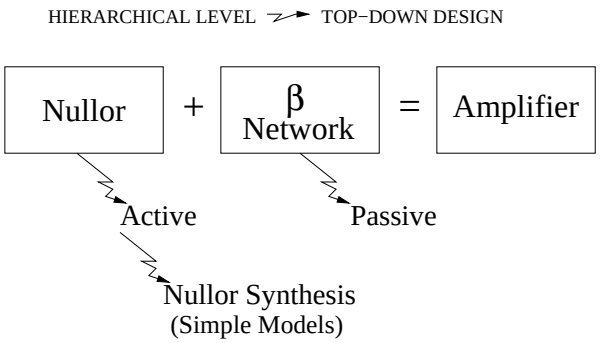

Figure 2.5: Amplifier scheme with nullor and feedback network.

The transfer of this system is given by:

$$A_t = \frac{A}{1 - A\beta} + A_{t0} \tag{2.4}$$

When the amplifying part is a nullor then loop gain is ∞, the gain is:

$$\lim_{n \to \infty} A_t = -\frac{\xi\nu}{\beta} + A_{t0} = A_{t\infty} \tag{2.5}$$

The gain A_t can be expressed in terms of $A_{t\infty}$ as:

$$A_t = A_{t\infty}\frac{-A\beta}{1 - A\beta} + \frac{A_{t0}}{1 - A\beta} \tag{2.6}$$

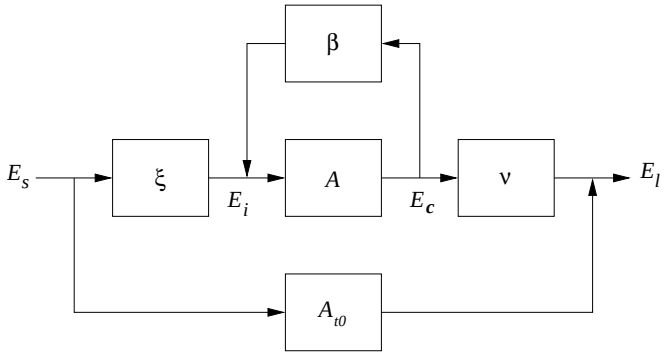

Figure 2.6: A negative feedback system.

The first factor ($A_{t\infty}$) is determined by the feedback network, its second factor is determined by the quality of the nullor implementation. Generally the second term $\frac{A_{t0}}{1-A\beta}$ can be neglected for reasonable loop gains. Nevertheless, this does not mean that A_{t0} is not important. It is also a term in $A_{t\infty}$, so even when a nullor is present, a direct transfer caused by the feedback network is not removed. With the aid of this model, it is possible to divide the synthesis of a negative feedback amplifier into two orthogonal parts. The ideal transfer $A_{t\infty}$ is designed under the assumption that a nullor is present and it remains unchanged during the design of the nullor.

2.3.2 Top-Down Approach

The highest level of the synthesis is shown in Figure 2.7.

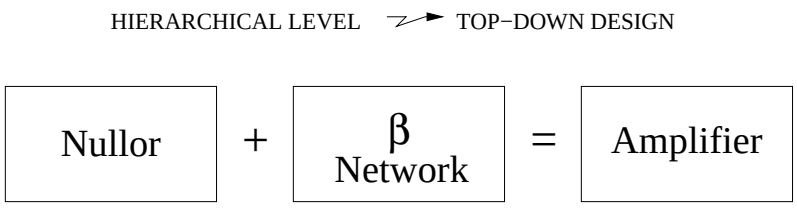

Figure 2.7: Highest Level of Synthesis.

As previously stated, this scheme is going to be implemented by active devices for the nullor synthesis and passive devices for the feedback network synthesis. This leads that the nullor synthesis is going to be performed according to the Figure 2.8. The flow diagrams for the three stages are given in figures 2.9, 2.10 and 2.11.

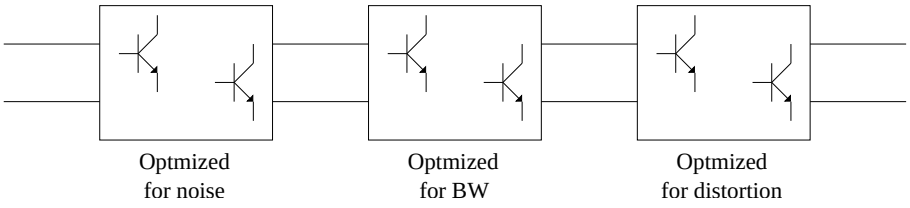

Figure 2.8: Nullor Synthesis.

2.3.2.1 Basic Configurations

Figure 2.5 showed the highest level for the amplifier design. The next step for amplifier design is to provide configurations with passive devices, i.e. lowering one level. For the basic configurations only one loop for feedback is taken. There are four basic configurations which are related to the four parameters that form Chain-matrix. Due to nullor properties only one transmission parameter applies to each configuration:

- Voltage Amplifier

$$\begin{pmatrix} u_i \\ i_i \end{pmatrix} = \begin{pmatrix} \frac{R_1}{R_1+R2} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u_o \\ i_o \end{pmatrix} \tag{2.7}$$

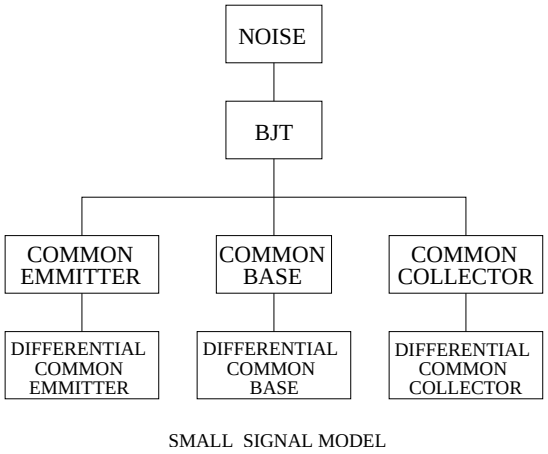

Figure 2.9: Noise Stage Synthesis.

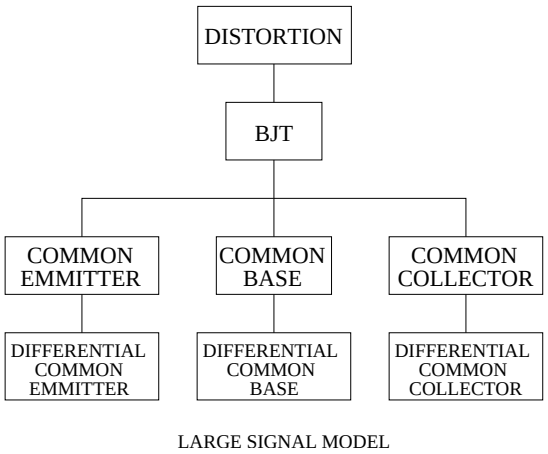

Figure 2.10: Distortion Stage Synthesis.

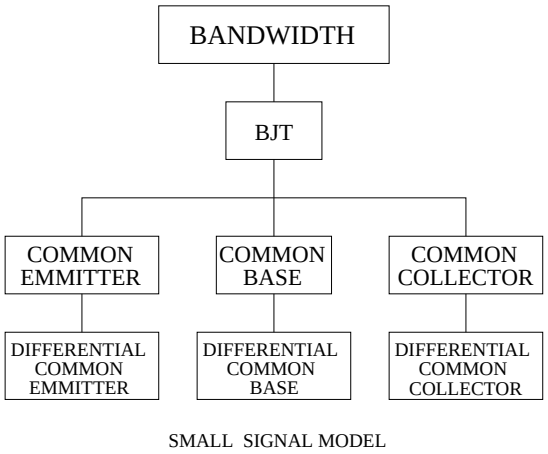

Figure 2.11: BW Stage Synthesis.

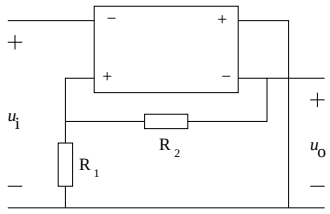

Figure 2.12: Single loop nullor based voltage amplifier.

its scheme is given in Figure 2.12.

- Transadmittance Amplifier

$$\begin{pmatrix} u_i \\ i_i \end{pmatrix} = \begin{pmatrix} 0 & -R \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u_o \\ i_o \end{pmatrix} \tag{2.8}$$

its scheme is given in Figure 2.13.

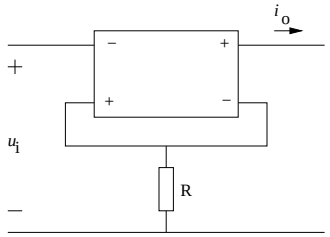

Figure 2.13: Single loop nullor based transadmittance amplifier.

- Transimpedance Amplifier

$$\begin{pmatrix} u_i \\ i_i \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ -\frac{1}{R} & 0 \end{pmatrix} \begin{pmatrix} u_o \\ i_o \end{pmatrix} \tag{2.9}$$

its scheme is given in Figure 2.14.

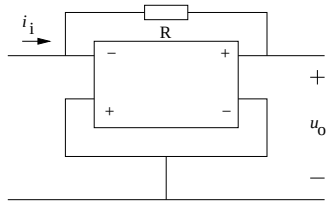

Figure 2.14: Single loop nullor based transimpedance amplifier.

- Current Amplifier

$$\begin{pmatrix} u_i \\ i_i \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & \frac{R_1}{R_1+R_2} \end{pmatrix} \begin{pmatrix} u_o \\ i_o \end{pmatrix} \tag{2.10}$$

its scheme is given in Figure 2.15.

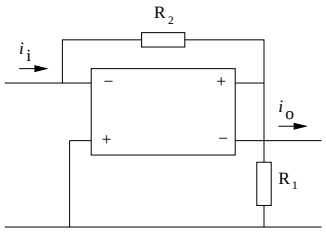

Figure 2.15: Single loop nullor based current amplifier

More complex topologies can be achieved as a result of the combination from these four basic topologies. The result of these combinations are two basic topologies but now for two loop amplifiers. The Chain-matrix for the two topologies now shows contributions for the four transmission parameters, i.e. it is possible to obtain all four kinds of transformations. Its Chain-matrix and schemes are given as follows:

- This new topology is a combination of single loop voltage amplifer and current amplifier, this is referred as *Two-loop (A)*. Its Chain-matrix is given by:

$$\begin{pmatrix} u_i \\ i_i \end{pmatrix} = \begin{pmatrix} \frac{R_2}{R_1+R_2} & 0 \\ \frac{R_2+R_3}{(R_1+R_2)(R_3+R_4)} & \frac{R_3}{R_3+R_4} \end{pmatrix} \begin{pmatrix} u_o \\ i_o \end{pmatrix} \tag{2.11}$$

in (2.11) the term for transadmittance is equal zero, therefore the transadmittance amplifier factor is ∞. The latter means that it is not possible to control this gain factor thus not possible to be implemented. Its scheme is given in Figure 2.16.

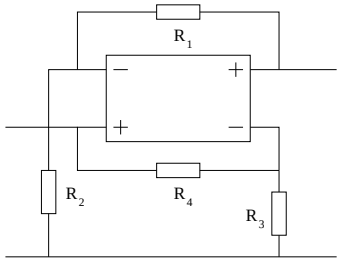

Figure 2.16: Two-loop (A) nullor based amplifier.

- The second two loop new topology is a combination between the transimpedance and transadmittance amplifers, this is referred as *Two-loop (B)*. Its Chain-matrix is as follows:

$$\begin{pmatrix} u_i \\ i_i \end{pmatrix} = \begin{pmatrix} -\frac{R_2}{R_1-R_2} & -\frac{R_1R_2}{R_1-R_2} \\ -\frac{1}{R_1-R_2} & -\frac{R_2}{R_1-R_2} \end{pmatrix} \begin{pmatrix} u_o \\ i_o \end{pmatrix} \tag{2.12}$$

all four elements in (2.12) provide a negative phase with respect to the phase of the signal at the input. Here it is possible to control gains by the adequate selection for the passive devices. Its scheme is presented in Figure 2.17.

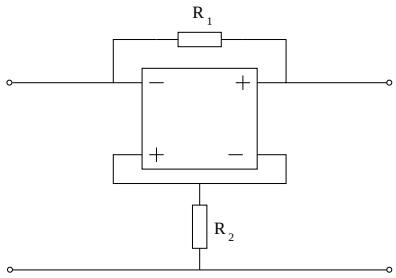

Figure 2.17: Two-loop (B) nullor based amplifier.

2.3.2.2 Noise Stage

This is the first block to be designed on the design process based on structured design to automate. An algorithm has been designed in such a way to guarantee that not matter which one of the single loop or two loop amplifier structures the noise level is kept within the value provided on specs. First thing to do is add the adequate noise sources to devices which comprise the topology with nullor. This sources could be noise voltage source or noise current source, it depends on the driving source to be amplified and the topology selected. If the driving source is voltage then all related noise sources for the devices to add are noise voltage sources; for the current case noise current sources are added. For the conductance case voltage noise sources are added and the transresistance amplifier current noise sources should be on the required locations.

Once the nullor synthesis is performed it is necessary to employ the noise model for the BJT transistor. This model is the small signal model adding one noise voltage source due to the internal base resistor and two current noise sources that models noise caused by physical process occurring at the base and collector.

The methodology for the noise stage consists on taking every noise source in the circuit and place them at the input port. This process is performed by using some transformations [33]:

Transformation 1. Voltage Source Shift.
This transformation allows to move the noise sources through the amplifier network. This source movement can not violate in any way the *Kirchhoff's voltage law* on any part of the circuit.

Transformation 2. Current source Shift.
Basically the same principle as the voltage shift applies here. Now it is necessary to do not violate the *Kirchhoff's current law*. This shift allows to move a current source around the network, but can not be used to disconnect it from the original nodes.

Norton-Thevenin Transform.
Helps to transform a noise current source into a noise voltage source and viceversa. Basically this kind of transformation does not move sources around the circuit.

Shift through Two-ports.
The transformation is established using the chain matrix of the two-port. The process

performed is to shift noise current and voltage sources located at the output port of a two-port circuit and place them at the input port. Transformed sources are affected on this way:

- Voltage source v_n at the output port, shifted to the input port transforms to: a) voltage source Av_n and b) current source Cv_n.
- Current source i_n at the output port, shifted to the input port transforms to: a) voltage source Bi_n and b) current source Ci_n.

where the subscript n refers that it is a noise source.

Once all noise sources movements to the input port are completed, the last step is to sum up all noise contributions generated by them. If the noise value is greater than expected then an adjustment of values should take place. After noise contribution is within parameters it is guaranteed that no matter how the remain stages are implemented, the noise is not going to be increased anyhow.

2.3.2.3 Distortion Stage

Noise stage is the first one to be synthesized and developed, on the other hand the distortion stage is the last block. This does not mean that it is not important, just that by applying the orthogonality principle the ideal stage to place it is at the output of the synthesized nullor. Distortion describes a deviation on the performance of the operating circuit in comparison to the desired behavior [1].

There are two kinds of distortion behaviors that are of main interest. Although they have the same origin their effects on the amplifier behavior are different, these are:

- Weak Distortion.
- Clipping distortion.

Weak Distortion

Occurs when the signal becomes so large that the higher order terms can not be disregarded but do not change entirely the signal behavior of the amplifier, this is called *weak distortion*. This effect can be disminished by an increase of the loop gain.

Clipping Distortion

When the signal becomes so big that the small-signal model is no longer valid such that the parameters of the small-signal model becomes highly independent of the signal is called *clipping*. This happens when the signal has very large amplitude. Any modification to the overall bandwidth caused by this last stage can be corrected elsewhere in the amplifier (orthogonality principle).

Gain on this stage must be as high as possible.

2.3.2.4 Bandwidth Stage

To design an amplifier to accomplish an exact bandwidth behavior is complicated and implies a lot of work. Consequently is necessary to perform some simple measurements in order to predict the bandwidth capabilities of the circuit for a given frequency range, this could help to decrease design time for this stage.

By means of the **Loop-gain-Poles** (*LP product*) is possible to perform really simple estimates related to the bandwidth handled by the amplifier. When the LP product is too low (this depends on the bandwidth specified) is **impossible** to reach that desired bandwidth. An increase on the LP product is a must.

LP product increase can be performed by two means:

- Increasing the LP product without increasing the order.
- Increasing the LP increasing the order.

After the LP product has the required value, now the location of the poles is of concern. This is because a transfer function of Butterworth type is desired. This process is detailed on the next chapter.

2.4 User Interface

This component of the CAD tool is important because it has a number of functions designed to allow the user to communicate with the rest of the system in a manner as close to his own language (for example technical English) as is possible [2]:

- The user interface (UI) will contain interpreters for translating the user's instructions and queries into machine usable instructions.
- Interpreters for the reverse direction i.e. for representing the results of the machine's processing in a presentable form.
- Mechanisms for explaining and justifying to the user the reasons why it came to a particular conclusion about a problem, i.e. providing a 'human window' onto the system (a term coined by Professor Donald Michie of Edinburgh University).

State of the art user interfaces developed as part of CAD frameworks must meet a wide set of requirements. These new interfaces must also display new forms of output. These include intermediate design data (in both graphic and non-graphic forms) [3]. At the lowest level, a graphics interface provides an abstraction that hides the details of the underlying graphics input and output devices. Early work in this area was done before the wide acceptance of multi-window interfaces like those developed at Xerox [47].

Complex user interfaces often consist of many windows each displaying different kinds of information and each responding to different kinds of user input. The basic graphic interface provides the functionality to implement such an interface but at a level that requires an overwhelming amount of programming. Much of this programming is common to most of

the applications. Toolkits, the next level in the architecture of modern user interfaces, have been developed to encapsulate the common portions of multi-window applications. These toolkits are based on the idea that complex interfaces can be built by combining standard components known as widgets. These components provide the labels, toggle switches, text input editors, menus, and other similar features found in multi-window interfaces [48], [49].

Major influences for future work in CAD framework interfaces is a need of standarization of toolkit and widget set functionality and by improvements to the programming interface. Standarization of toolkit and widget set functionality will allow framework user interface developers to make greater use of these systems and thus provide much more powerful interfaces than those in existence today. Since these systems become standarized, those developing other tools outside the real of the CAD frameworks will also begin producing systems meeting these specifications. The result is the possibility to increase the number of uniform computing environments where a designer uses CAD tools and non-CAD tools in an interchangeable fashion.

Early CAD systems required users to encode graphic information textually and interpret numeric output. Though the use of modern graphic interfaces, much less coding is involved. Future systems are going to continue this trend untilusers no longer care about the underlying hardware and software architectures used to implement the system. Instead, users will focus on the problem at hand, not on the tool they use to solve the problem [3].

2.5 Amplifier Design Automation

By now it has been established guidelines focused to decrease complexity problems related to the design problem. The basic model and basic topologies are explained as well. On this section the basics for automate amplifier design are presented.

Automation can be considered to be a knowledge transfer process, where knowledge is being transferred from the automating human to a computer system. Not all knowledge is equally suitable for such transferral.[33]

Not all human knowledge can be easily translated into computer language code. Programmers resort to flow diagrams, pseudo-code, and other tools in order to translate human knowledge that needs to be automated because complexity or a process is repeated indefinitely.

The analog electronic design automation can provide tools that anyone would be able to operate no matter if is a bachelor, master or PhD even a hobbyist could be capable to use them. These tools can be employed to teach basic electronics or to design state of the art designs. Since it is not possible to create a universal design tool, CAD tools development must focus on specific needs. The most important fact that these kind of tools must compply is to avoid as much as possible the "human guess", that is, provide the user with as much information as possible and in case of doubt offer supplemental information.

Structured design allows to divide design process in basic blocks, and each block can be subdivided as much as needed to simplify calculations. It is focused on automate a process which is to design amplifiers. In order to perform this amplifier design the user must supply some basic data in order to establish requirements that the design must fulfill, as a first step. Next step is to define what kind of source (current or voltage) will drive this design

and what kind of information is expected (voltage or current) at the output port. Providing this basic information it is possible to perform various calculations and provide a result but here is where the "human factor" takes an important role. The tool should let user to select certain active device. In that case it is not possible to achieve a successful result the user is notified and ask him/her to select another device. This process is repeated as many times as the programmer structured the program.

Other important fact to be taken into account is how the program is going to provide the final results. One thing for granted is that the results are displayed on the terminal screen but there are many other choices to provide results. It can be saved like an spice netlist, as a simple text file or some other format for some other CAD tool.

Another point provided by structured design is model simplicity which lets perform faster calculations but another question arises, how "simple" models should be programed?. An ambitious programmer could provide robust models but this may have an impact on the overall performance and goes against basic principles of structured design.

Many more questions must be answered as the program is being developed but one thing must be taken into account, software programmers must have "electronics" approach. This means that is not possible in any way the programmer creates such a robust tool that it ends up more oriented for computer science users than a useful CAD tool.

2.6 Conclusions

Here were shown the basic steps to perform design based on structured design. The main objective of this approach is to provide guidelines toward a faster development but also gives foundations to automate this process. The asymptotic gain model has proven to be ideal for developing negative feedback amplifiers.

The basic block on which this theory is based was also presented, the nullor. This ideal device is formed by a nullator at the input port and a norator at the output port. Its ideal state was proven showing its Chain-matrix. An ideal amplifier is done by a nullor and a feedback network.

It was presented basic configurations for one and two loop amplifiers and how its gains can be controlled by selecting adequate values for the feedback network. The way that each basic design aspects are performed are explained as follows: noise stage design (Chapter 3), clipping distortion stage design (Chapter 4) and bandwidth stage design (Chapter 5). The overall CAD tool development is presented in Chapter 6 and some design examples based on the design tool are presented in Chapter 7.

Chapter 3

Noise Stage

3.1 Introduction

The electrical noise sources in passive elements and active electronic devices have been investigated extensively, and appropriate models have been derived [50, 51, 52, 53]. The noise performance of an electronic circuit ordinarily can be analysed in terms of these models by considering each of the uncorrelated noise sources in turn and separately computing its contribution at the output. Nevertheless there is a methodology that shows how to take each and every noise source in the amplifier structure (Figure 3.1) and move them right at the input port (Figure 3.2), by this way it is possible to obtain an equivalent noise source placed at the input port.

Noise is treated in terms of the equivalent input noise sources rather than by noise figure, which is less efficient and often confusing. First, in a qualitative sense, noise figure and noise factor are the same [54], and in casual conversation they are often interchanged. Second, the concept of noise factor has three major limitations [54]:

1. Increasing the source resistance may decrease the noise factor while increasing the total noise in the circuit.

2. If a purely reactive source is used, noise factor is meaningless since the source noise is zero, making the noise factor infinite.

3. When the device noise is only a small percentage of the source thermal noise (as with some low noise FETs), the noise factor requires taking the ratio of two almost equal numbers. This can produce innacurate results.

The noise performance of amplifiers, besides being dependent on the amplifier, is also a function of the signal source impedance and frequency range; these two factors determine the optimum input stage [55]. The noise performance of FET's at low frequencies is related to g_m, and at high frequencies to f_T. Low-noise junction transistors should have high current gain β, a minimum base resistance r_b and a high cutoff frequency f_T. For resistive signal sources and low-frequency inductive sensors, "ideal" amplifiers can be designed where the added noise is negligible in comparison to the inherent thermal noise generated in the source.

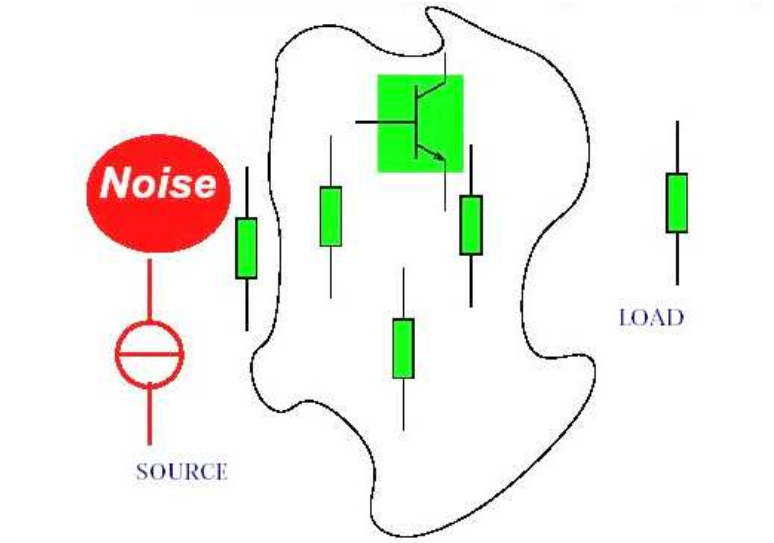

Figure 3.1: Overall noise sources.

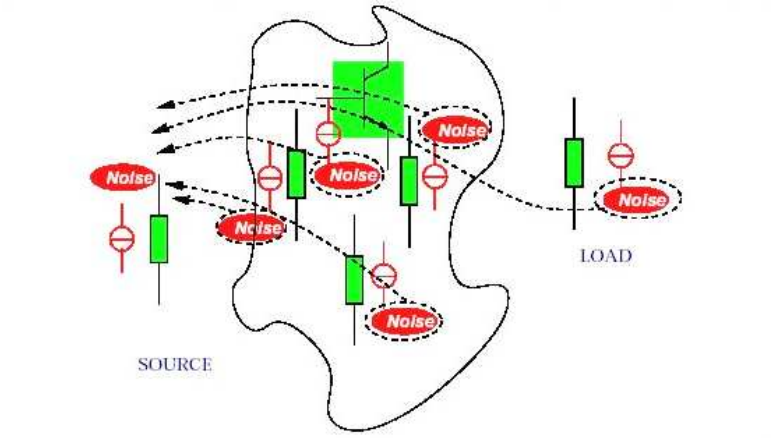

Figure 3.2: Moving noise sources to the input port.

In the following sections the most common devices and its noise sources are presented, an insight of the noise design process is provided and this chapter concludes giving a general algorithm to accomplish a desired noise level.

3.2 The Chain-Matrix

There are many ways to describe a two-port circuit, but this work is based on the Chain-matrix methodology. This is the best way to describe a system with cascaded stages and every stage within the system is describe by a Chain-matrix also. Chain-matrix relates the input voltage and current of a two-port system to the output voltage and current [1]. It is represented as follows:

$$\begin{pmatrix} u_i \\ i_i \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} u_o \\ i_o \end{pmatrix} \tag{3.1}$$

with:

$$A = \frac{1}{\mu} = \left.\frac{u_i}{u_o}\right|_{i_o=0} \tag{3.2}$$

$$B = \frac{1}{\gamma} = \left.\frac{u_i}{i_o}\right|_{u_o=0} \tag{3.3}$$

$$C = \frac{1}{\zeta} = \left.\frac{i_i}{u_o}\right|_{i_o=0} \tag{3.4}$$

$$D = \frac{1}{\beta} = \left.\frac{i_i}{i_o}\right|_{u_o=0} \tag{3.5}$$

3.3 Power Spectral Density

The power spectral density is the key towards the calculation of the equivalent input noise power. According to its definition, the power spectral density should be obtained through Fourier transformation of the so called *autocorrelation function* denoted by $R(r)$ [1]:

$$S(f) \triangleq \int_{-\infty}^{\infty} R(r) exp(-j2\pi f\tau) d\tau \tag{3.6}$$

The autocorrelation function expresses "how fast a stochastic process fluctuates in time on average". Since for physical stochastic processes $R(r)$ is a symmetrical real-valued function, $S(f)$ is also real valued (and symmetrical around $f = 0$).

It is possible to circumvent the elaborate calculation of the autocorrelation function and the Fourier transformation. This can be done due to the fact that the power spectral density spectra of the various "uncorrelated" circuit noise processes are generally known. Then it is only necessary to express the power spectral density of the equivalent input noise in terms of the spectral densities of the various noise processes. Applying the Wiener-Kintchine

theorem which states that when a noise process $e_{n,y}(t)$ equals the convolution of a linear-time invariant impulse-response $h(\tau)$ and a noise process $e_{n,x}(t)$:

$$e_{n,y} = h * e_{n,x}(t) \tag{3.7}$$

then its power spectral density $S_{n,y}(f)$ is related to the power spectral density $S_{n,x}(f)$ of $e_{n,x}(t)$ through

$$S_{n,y}(f) = |H(j2\pi f)|^2 S_{n,x}(f) \tag{3.8}$$

where the transfer function $H(j2\pi f)$ is the Fourier transform of $h(\tau)$. Therefore the power spectral density of the equivalent input noise can be expressed as:

$$S_{n,eq,in}(f) = \sum_{i=1}^{n} |H_{2i}(j2\pi f)|^2 S_{v_{n,2i}(f)} + \sum_{k=0}^{m} |H_{2k+1}(j2\pi f)|^2 S_{i_{n,2k+1}}(f) \tag{3.9}$$

where it is assumed that all involved noise processes are uncorrelated.

Within next subsections, the power spectral density of an electrical noise process is often given in terms of the mean-square value of the noise related to a frequency band $[f, f + \Delta f]$. This value is the power contained in a frequency band Δf around frequency f. Consequently if $\overline{v_n^2}(f, \Delta f$ denotes the mean-square value of a noise voltage $v_n(t)$ in a frequency band Δf around f, and $S_{v_n}(f)$ denotes the associated power spectral density, then the two are related through:

$$S_{v_n}(f) = \lim_{\Delta f \to 0} \frac{\overline{v_n^2}(f, \Delta f}{\Delta f}. \tag{3.10}$$

3.4 Noise Types

Since the noise generated by a device depends on the material which is made of and the physics of such device, there are several noise types. The must common are presented on the following paragraphs.

3.4.1 Thermal Noise

Noise in electronic devices can be attributed to two main processes: thermal noise and shot noise. As a result of thermal fluctuation of charge carriers a noise voltage can be measured in series with a resistor of a value R, whose power density is

$$\overline{e^2}_n = 4kTR \quad (V^2/Hz) \tag{3.11}$$

where T is the absolute temperature of the resistor and $k = 1.38X10^{-23}\ J/K$ is Boltzmann constant. This density is constant to frequencies up in the infrared, where it begins to drop due to quantum-mechanical effects [55]. It follows, an actual resistor can be represented by a noiseless resistor in series with a voltage source. Or, equivalently, by a parallel current noise source using the Thevenin-Norton transform, which gives

$$\overline{i^2}_n = \frac{4kT}{R} \quad (A^2/Hz) \tag{3.12}$$

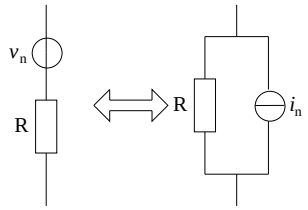

Figure 3.3: Thermal noise in resistors.

The generation of thermal noise is ideally not affected by the flow of current through the resistor. However, carbon resistors, in particular, have an additional current dependent noise which makes them unsuitable for critical applications. In contrast to resistors, ideal capacitors and inductors do not generate noise. For a complex impedance $Z(\omega) = R(\omega) + jX(\omega)$ the spot value of the thermal noise will be due to resistor $R(\omega)$, and its density will, thus, be frequency dependent.

3.4.2 Shot Noise

Since electric current is composed of discrete charge carriers, fluctuations are present in the current crossing a barrier where the charge carriers pass independently of one another. Examples are: the p-n junction diode where electron emission occurs as a result of thermal motion, and photodiodes where the absorption of photons is involved. This effect does not exist, for example, in metallic conductors because of long-range correlation between charge carriers. The fluctuations manifest themselves as a noise component named *shot noise* [1, 55], which can be represented by an appropriate current source in parallel with the dynamic impedance of the barrier across which it is generated.

3.4.3 Generation-Recombination Noise

Photoconductive detectors produce *generation-recombination (GR) noise* [55] in response to a steady irradiance. Hole-electron pairs are generated randomly and recombine randomly by a statistically unrelated process. Thus full GR noise neglecting that which is thermally generated, corresponds to twice the shot noise on the absorbed background photon rate for intrinsic photoconductors. As the DC bias is increased, a voltage is reached at which the minority carrier (hole) transit time is less than the lifetime. At this bias, the carriers are swept out of the device before they recombine and the GR noise approaches shot noise on the photocurrent.

3.4.4 Flicker Noise

Semiconductor devices as well as vacuum-tubes show, an additional noise component that is inversely proportional to frequency. Hence, it has the name $1/f$ *noise* (also referred to

as excess, flicker, or pink noise). This noise in semiconductors is associated, mainly, with crystal surface conditions. It occurs, however, in nonelectrical phenomena, as well [56]. When associated with current noise, it can be described by the spectral density

$$\overline{i_n^2} = \overline{i^2{}_{n0}}\left(1 + \frac{f_L}{f^n}\right) \tag{3.13}$$

$\overline{i^2{}_{n0}}$ represents the white shot noise component, in the excess noise component f_L is the empirical value of the break frequency where the two noise components are equal and is usually subject to process spreads. The actual value of n is not necessarily fixed with frequency. However, in junctions transistors it is in the vicinity of 1.1. In some operational amplifiers this value was verified down to 10^{-7} Hz [57]. Apparently, this may lead to very high noise amplitude. However, for the type of spectral density in (3.13) the noise power in each frequency *decade* is approximately constant. In practice, flicker noise is often regarded as a DC instability.

3.5 Noise Models for Electronic Devices

The three important types of noise in electronic circuits are shot noise, thermal noise and flicker noise. The noise models for resistors, junction diodes and bipolar junction transistors are presented.

3.5.1 Resistor Noise Model

As mentioned in 3.4.1, the noise model for a resistor can be given in two ways: as a noise voltage source in series with a noiseless resistor or a shunt noise current source with a noiseless resistor. Nevertheless, Nyquist has shown that the voltage fluctuations (voltage noise) observed at the terminals of a resistor with value R due to thermal agitation possesses a mean-square value associated to a frequency band $\Delta f = (f_2 - f_1)$ (where f_2 is the upper cutoff frequency and f_1 is the lower cutoff frequency) equal to [1]:

$$\overline{u^2{}_n}(f, \Delta f) = 4kTR\Delta f, \quad f \geq 0 \tag{3.14}$$

and the noise current for a frequency band Δf is:

$$\overline{i^2{}_n}(f, \Delta f) = \frac{4kT\Delta f}{R}, \quad f \geq 0 \tag{3.15}$$

3.5.2 Diode Noise Model

Diodes, whether pn-junction or Schottky, produce *shot noise*. Figure 3.4 depicts the noise model of a diode. The mean square value of the noise current is proportional to the average external diode current I_d, that is, the diode bias current:

$$\overline{i^2{}_n}(f, \Delta f) = 2qI_d\Delta f, \quad f > 0 \tag{3.16}$$

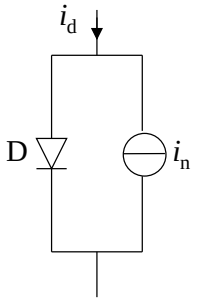

Figure 3.4: Noise model for diode.

3.5.3 Bipolar Noise Model

Bipolar transistors produce a combination of shot noise, thermal noise and flicker noise. The region of interest is the linear operating region for the device, on this region the noise is determined by four uncorrelated noise processes. Shot noise are generated by two sources: the first source is created by the diffusion current injected from the emitter into the base, known as the collector current I_c, is transported out of the base through the reverse biased *base-collector* junction. The second source is created by the diffusion current injected from base into emitter, known as base current I_b, generating a shot noise source connected between *base* and *emitter*.

Flicker noise, due to process imperfections, generates a source which is located between *base* and *emitter*. Nevertheless, for modern transistors the frequency f_L is usually very small (up to a few Hz [1]), such that $1/f$ noise is mostly negligible.

Thermal noise is produced by the resistance of the *bulk* material connected to the intrinsic *base-emitter* and *base-collector* junctions. Often referred as r_b, the value of *bulk* resistance of *base* is generally greater than *collector* and *emitter* bulk resistances, therefore producing non-negligible noise.

Equations for the shot noise (i_b, i_c), flicker noise (i_{bf}) and thermal noise (u_{nb}) are given, respectively, as follows:

$$\overline{i^2{}_c}(f, \Delta f) = 2qI_c\Delta f, \quad f \geq 0 \tag{3.17}$$

$$\overline{i^2{}_b}(f, \Delta f) = 2qI_b\Delta f, \quad f \geq 0 \tag{3.18}$$

$$\overline{i^2{}_{bf}}(f, \Delta f) = 2qI_b\frac{f_L}{f}\Delta f \tag{3.19}$$

$$\overline{u^2{}_{nb}}(f, \Delta f) = 4kTr_b\Delta f, \quad f \geq 0 \tag{3.20}$$

3.6 Structured Design Noise Guidelines

The structured design methodology stablishes that the first step in the noise analysis is to determine the so called *equivalent input noise source*. This source models the noise experienced at the output port of the amplifier due to internal circuit noise production by devices employed in the nullor synthesis. To determine the overall contribution for all internal noise sources into equivalent noise sources, it is necessary to resort to circuit theory.

3.6.1 Voltage Source Shift

The voltage source shift (V-shift) is a transformation that makes possible to move voltage sources, in this case noise sources, through the network that comprises the amplifier. It is employed to move these sources to the output, input or branch of the amplifier. Once this operation is performed, the source can be subject of another kind of transformation.

The major constraint to be assessed is that this movement can not change the *Kirchhoff Voltage Law (KVL)* of any maze within the circuit. An example is shown in Figure 3.5. The original noise source v_n is shifted out of the branch between the nodes 1 and 2 into the branch with nodes 2 and 3 and the branch with nodes 2 and 4. In order to guarantee that there is no issue with the KVL of *mazes I, II, III*, the new noise sources v_{n1} and v_{n2} have to be exactly equal to each other on the the original source v_n. In case that node 2 is connected to n branches, that is $n > 2$, the shifting of v_n through node 2 from its original branch to the $n-1$ other branches yields $n-1$ identical sources.

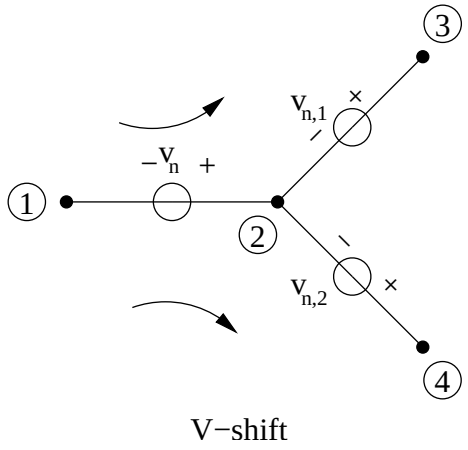

Figure 3.5: V-shift transform.

3.6.2 Current Source Shift

The dual transformation of the V-shift is the I-shift. This shift allows to move noise current sources through the amplifier network. Like V-shift, I-shift is employed to shift sources to the input port, output port or intermediate nodes within amplifier. After this operation is applied the resultant source allows another type of transformation.

Just like V-shift can not affect the KVL of any circuit maze, I-shift can not change the *Kirchhoff Current Law (KCL)* of any node within the circuit. An example is depicted in Figure 3.6. Here the original noise current source i_n is transformed from the branch between nodes 1 and 2 into sources i_{n1} and i_{n2} located between nodes 1,3 and 2,3 respectively. To keep KCL unchanged for nodes 1,2 and 3, sources i_{n1} ,i_{n2}, i_n have to be exactly equal to each other and be directed as illustrated. This shift allows to move a current source around the network but cannot be used to disconnect it from the original nodes.

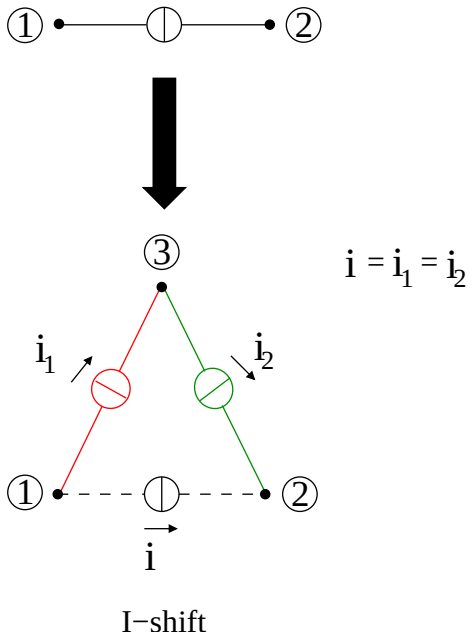

Figure 3.6: I-shift transform.

3.6.3 Norton-Thevenin Transformation

The equivalence of the well-known theorems of Norton and Thevenin can be used to transform a noise current source into a noise voltage source and viceversa. This type of transformation does not let move sources through the network but to switch between the V-shift and I-shift. This transformation is shown in Figure 3.7.

The current source i_n and the voltage source v_n hae a one to one correspondence through the impedance Z. The stochastic process i_n and v_n are therefore fully correlated and possess similar but not identical stochastic properties. This transformation does change the KVL's and KCL's of the circuit because it exchanges a circuit branch for a circuit node and vice versa. This is the reason why Norton-Thevenin transformation combined with the V-shift and I-shift can be employed to eliminate a voltage source from a maze or a current source from a node.

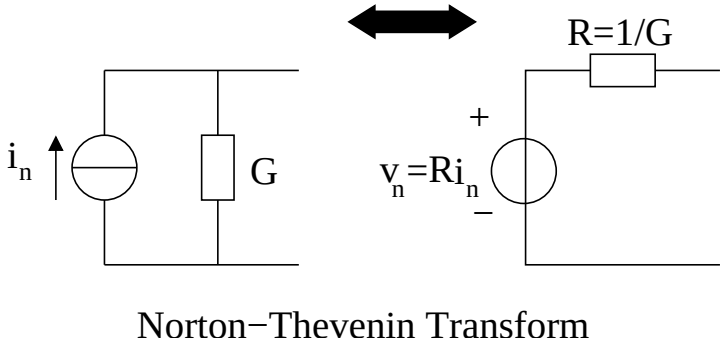

Figure 3.7: Norton-Thevenin transform.

3.6.4 Two-port Transformation

The previous three transformations are all concerned with two-terminal elements, that is one-port, only though. An amplifier network is a two-port implementation because the nullor or controlled sources are devices that cannot be implemented by combination of one-port devices. This is the main reason to employ an additional transformation, *the two-port transformation*. The transformation is illustrated in Figure 3.8.

The output voltage v_o and output current i_o of the two-port are affected by a noise voltage source v_n and a noise current source i_n respectively, as it can be seen on the upper part of the figure. The main purpose of the two-port transformation is to obtain the equivalent input noise sources for the two given output noise sources, lower part of Figure 3.8.

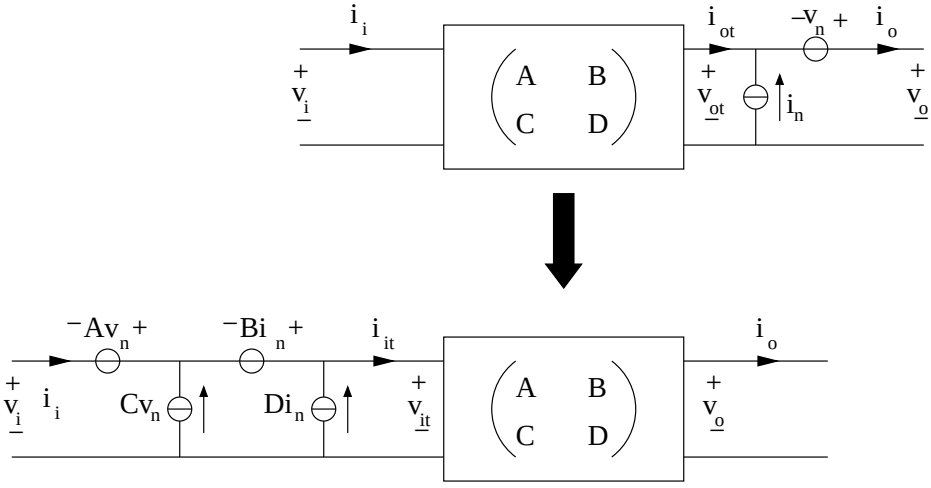

Figure 3.8: Two-port transformation.

3.7 Equations for Noise Calculations

Provided the necessary theory to obtain the representation for the input noise sources, it is possible to obtain the equivalent input noise formulas for the topologies and device configurations to be employed during the development of the design process.

3.7.1 Equations for Nullor Amplifier Topologies

Single loop topologies and two double loop topologies were provided in Chapter 2. The equations to be developed are for the *four basic topologies* and *two loop (B)*.

3.7.1.1 Voltage Amplifier Noise Equations

The voltage amplifier including noise sources is depicted in Figure 3.9. u_{ns} represents the voltage noise source of the signal source, u_{n1} and u_{n2} are noise sources generated by the

feedback implementation and u_n and i_n are noise sources generated by the nullor implementation of the first stage, i.e., the noise emanating from the synthesis of the first stage. The signal source is represented by u_s.

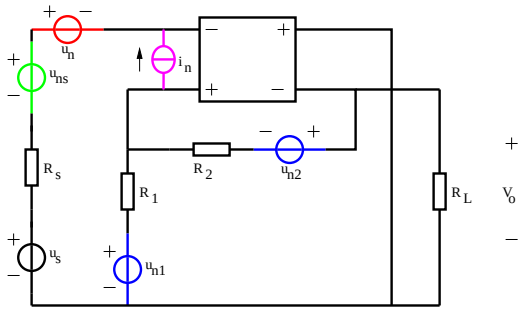

Figure 3.9: Nullor based voltage amplifier including noise sources.

Step 1

Since this is a voltage amplifier the equivalent noise source to be obtained is a voltage noise source. It is necessary to reduce as many sources as possible. The first sources to be reduced are u_{ns} and u_n, the resultant source is referred as $u'_{ns} = u_{ns} - u_n$. Figure 3.10 shows the new source.

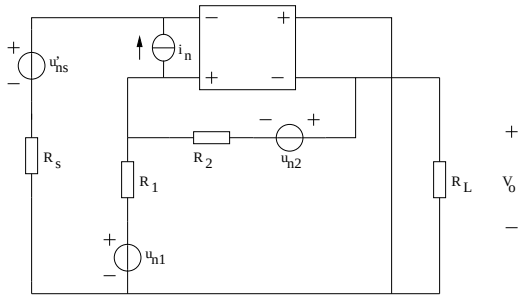

Figure 3.10: Nullor based voltage amplifier. Step 1.

Step 2

Now by means of an I-shift the noise current source placed at the input port of the nullor is converted in two noise current sources located at the input port of the amplifier. These sources are then transformed in voltage sources by means of a Norton-Thevenin transformation. One noise current source is transformed in a voltage source using the shunt resistance R_s while the other is transformed using the shunt resistance R_1. Figure 3.11 presents the topology.

Step 3

Again noise sources are reduced. Now u'_{ns} and u_{n1} becomes (Figure 3.12):

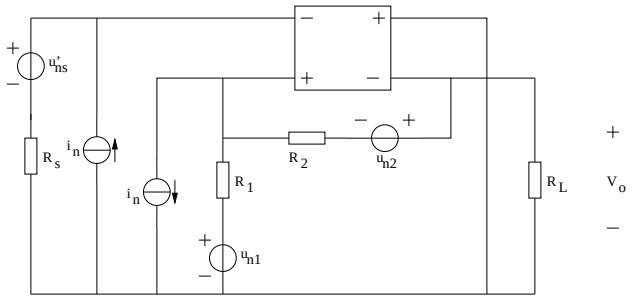

Figure 3.11: Nullor based voltage amplifier. Step 2.

$$u''_{ns} = u'_{ns} + R_s i_n \tag{3.21}$$

$$u'_{n1} = u_{n1} - R_1 i_n \tag{3.22}$$

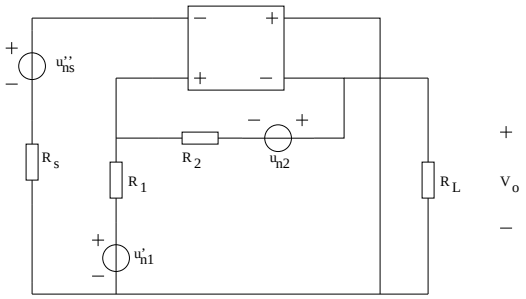

Figure 3.12: Nullor based voltage amplifier. Step 3.

Step 4

The next noise source to be moved is u'_{n1}. Since the branch where this source is located is connected to a joint of two branches, two noise sources are created in order to keep the KVL as seen previously. The first source created is located at the input branch while the other is "heading" to the output port. The latter source has to be placed at the output branch. Again a V-shift is applied to this source and once again this branch is connected to a joint of two branches and consequently two sources are created. However one of the sources is directly connected to one of the nullor's output port and because of nullor's property this noise source can be disregarded. The other source is located at the load branch and no further transformation is needed. Figure 3.13.

Step 5

At input port there are two noise voltage sources in series therefore they must be reduced, becoming:

$$u'''_{ns} = u''_{ns} - u'_{n1} \tag{3.23}$$

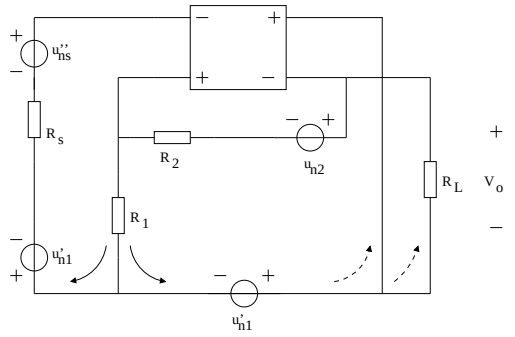

Figure 3.13: Nullor base voltage amplifier. Step 4.

The last noise voltage source to be shifted is u_{n2}. This step is simple because the branch is connected to a two branch joint, but one of the branches is the negative output of the nullor therefore it is not possible to be affected by this source. Finally this source is in series with u'_{n1}. Figure 3.14.

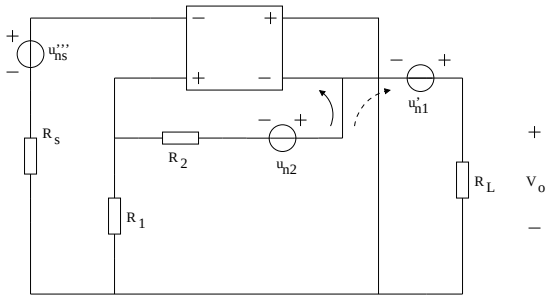

Figure 3.14: Nullor based voltage amplifier. Step 5.

Step 6

The output noise sources are reduced to (Figure 3.15):

$$u'_{n2} = u_{n2} + u'_{n1} \tag{3.24}$$

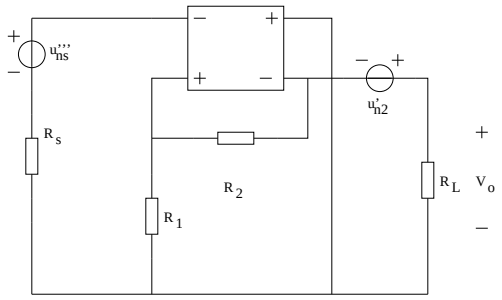

Figure 3.15: Nullor based voltage amplifier. Step 6.

Step 7

By means of a two-port transformation the u'_{n2} source is placed at the input port (Figure 3.16). The Chain-matrix for this amplifier is:

$$\begin{pmatrix} u_i \\ i_i \end{pmatrix} = \begin{pmatrix} \frac{R_1}{R_1+R2} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u_o \\ i_o \end{pmatrix} \tag{3.25}$$

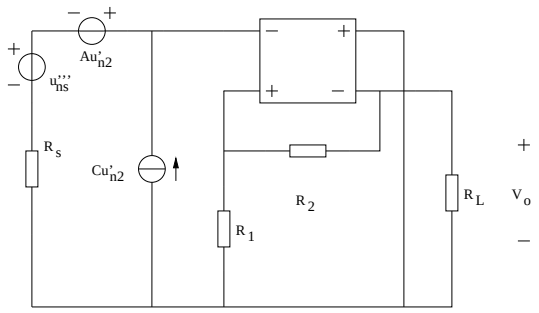

Figure 3.16: Nullor based voltage amplifier. Step 7.

as can be seen from the image the resultant current source has no effect on the overall noise behaviour because the Chain-matrix value is 0. Reducing all noise voltage sources gives for the spectral noise density:

$$S_{v_{n,eq,in}} = u_n + u_{ns} + R_s^2 i_n + \left(\frac{R_1 R_2}{R_1 + R_2}\right)^2 i_n + \left(\frac{R_2}{R_1 + R_2}\right)^2 u_{n1} + \left(\frac{R_1}{R_1 + R_2}\right)^2 u_{n2} \tag{3.26}$$

3.7.1.2 Transadmittance Amplifier Noise Equations

The way noise sources are moved in order to obtain an equivalent single noise source is detailed in Appendix A, subsection A.1.1. Figure 3.17 shows the noise sources for this topology. The spectral noise density for this amplifier is given by:

$$S_{v_{n,eq,in}} = u_n + u_{ns} + (R_S + R)^2 i_n + u_{n_R} \tag{3.27}$$

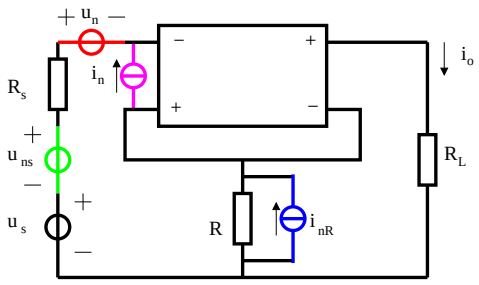

Figure 3.17: Transadmittance amplifier with noise sources added.

3.7.1.3 Transimpedance Amplifier Noise Equations

The way noise sources are moved in order to obtain an equivalent single noise source is detailed in Appendix A, subsection A.1.2. Figure 3.18 depicts the amplifier including its noise sources. The spectral noise density for this amplifier is given by:

$$S_{i_{n,eq,in}} = i_n + i_{ns} + \left(\frac{1}{R_S} + \frac{1}{R}\right)^2 u_n + i_{n_R} \tag{3.28}$$

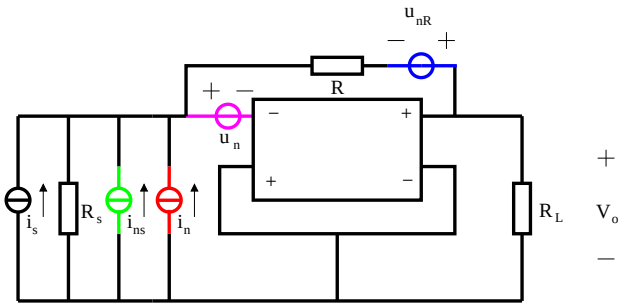

Figure 3.18: Transimpedance amplifier including noise sources.

3.7.1.4 Current Amplifier Noise Equations

The way noise sources are moved in order to obtain an equivalent single noise source is detailed in Appendix A, subsection A.1.3. Figure 3.19 shows the noise sources applied to the circuit. The spectral noise density for this amplifier is given by:

$$S_{i_{n,eq,in}} = i_n + i_{ns} + \left(\frac{1}{R_S} + \frac{1}{R_1 + R_2}\right)^2 u_n + \left(\frac{R_1}{R_1 + R_2}\right)^2 i_{n_{R_1}} + \left(\frac{R_2}{R_1 + R_2}\right)^2 i_{n_{R_2}} \tag{3.29}$$

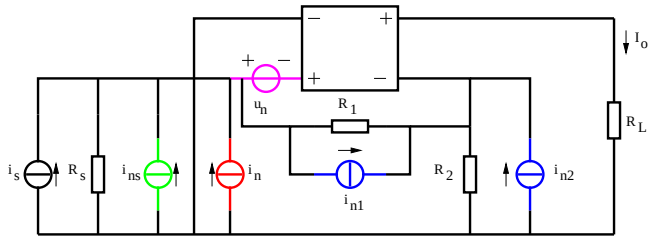

Figure 3.19: Current amplifier with noise sources.

3.7.1.5 Two-loop (B) V-V Amplifier Noise Equations

The way noise sources are moved in order to obtain an equivalent single noise source is detailed in Appendix A, subsection A.2.1. Figure 3.20 depicts the amplifier and its noise sources. The spectral noise density for this amplifier is given by:

$$S_{v_{n,eq,in}} = u_n + u_{ns} + \left(\frac{R_S R_2 + R_1 R_2}{R_1 - R_2}\right)^2 i_n + \left(\frac{R_S + R_2}{R_1 - R_2}\right)^2 u_{n_{R_1}} + \left(\frac{R_S + R_1}{R_1 - R_2}\right)^2 u_{n_{R_2}} \quad (3.30)$$

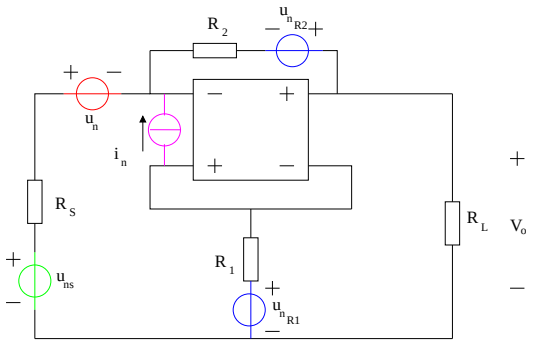

Figure 3.20: Two-loop (B) voltage to voltage amplifier with noise sources.

3.7.1.6 Two-loop (B) V-I Amplifier Noise Equations

The way noise sources are moved in order to obtain an equivalent single noise source is detailed in Appendix A, subsection A.2.2. The topology including the noise sources is shown in Figure 3.21. The spectral noise density for this amplifier is given by:

$$\begin{aligned} S_{v_{n,eq,in}} = u_n + u_{ns} + \left(\frac{R_S R_1 + R_1 R_2}{R_1 - R_2}\right)^2 i_n + \left(\frac{R_S R_2 + R_1 R_2}{R_1 - R_2}\right)^2 i_{n_{R_1}} \\ + \left(\frac{R_S R_1 + R_1 R_2}{R_1 - R_2}\right)^2 i_{n_{R_2}} \end{aligned} \quad (3.31)$$

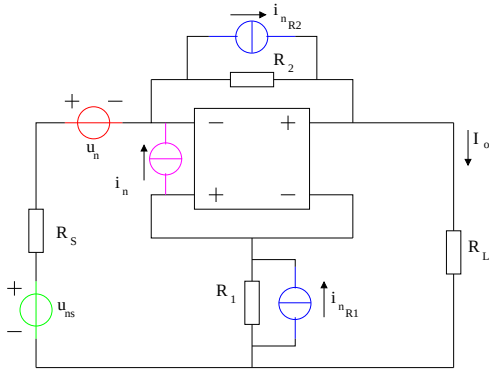

Figure 3.21: Two-loop (B) voltage to current amplifier including noise sources.

3.7.1.7 Two-loop (B) I-V Amplifier Noise Equations

The way noise sources are moved in order to obtain an equivalent single noise source is detailed in Appendix A, subsection A.2.3. Figure 3.22 depicts the noise sources added to the topology. The spectral noise density for this amplifier is given by:

$$S_{i_{n,eq,in}} = i_n + i_s + \left(\frac{R_S + R_2}{R_S R_1 - R_S R_2}\right)^2 u_n + \left(\frac{R_S + R_2}{R_S R_1 - R_S R_2}\right)^2 u_{n_{R_1}} + \left(\frac{R_S + R_1}{R_S R_1 - R_S R_2}\right)^2 u_{n_{R_2}} \tag{3.32}$$

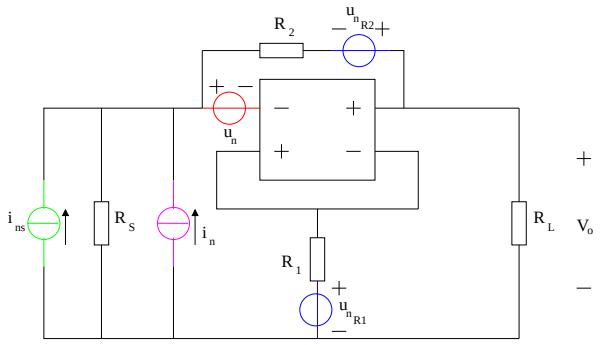

Figure 3.22: Two-loop (B) current to voltage amplifier and noise sources.

3.7.1.8 Two-loop (B) I-I Amplifier Noise Equations

The way noise sources are moved in order to obtain an equivalent single noise source is detailed in Appendix A, subsection A.2.4. The noise sources are added to the topology and can be appreciated in Figure 3.23. The spectral noise density for this amplifier is given by:

$$S_{i_{n,eq,in}} = i_n + i_s + \left(\frac{R_2 + R_S}{R_S R_1 - R_S R_2}\right)^2 u_n + \left(\frac{R_S R_2 + R_1 R_2}{R_S R_1 - R_S R_2}\right)^2 i_{n_{R_2}} + \left(\frac{R_S R_1 + R_1 R_2}{R_S R_1 - R_S R_2}\right)^2 i_{n_{R_1}} \tag{3.33}$$

3.8 Equations for BJT Configurations

At this point the noise stage only comprises the noise produced by the passive devices. Active devices are chosen to *perform* like a nullor. In this case the employed active device is the BJT. As stated on pevious paragraphs this device has more than one noise source. For all preceeding equivalent noise equations the noise sources generated by the nullor synthesis are already taken into accont. Nevertheless, is not possible to provide just one equation that encompasses the noise behaviour for all the BJTs devices since it varies on each configuration. The noise equations for all configurations are expressed within the next subsections.

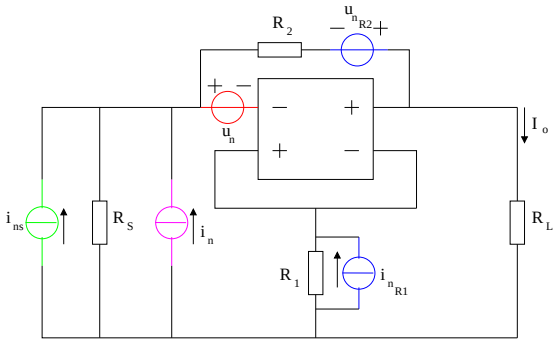

Figure 3.23: Two-loop (B) current to current amplifier including noise sources.

3.8.1 Common Emitter

The Chain-matrix for a BJT device in common emitter configuration is:

$$K_{CE} = \begin{pmatrix} -\frac{1}{r_o g_m} & -\frac{1}{g_m} \\ -\frac{1}{r_o \beta} & -\frac{1}{\beta} \end{pmatrix} \tag{3.34}$$

Figure 3.24 shows where noise sources are located.

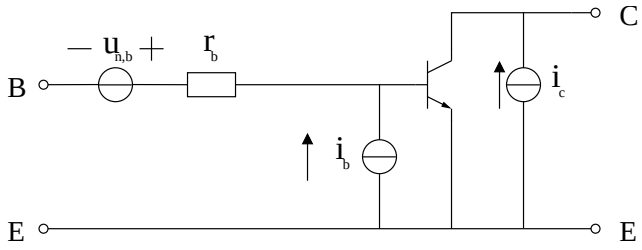

Figure 3.24: Noise sources for BJT in Common Emitter.

It is an easy chore to find out the equivalent noise sources for this configuration. By means of a two-port transformation the source i_c is placed "at the input port" of the device. Figure 3.25.

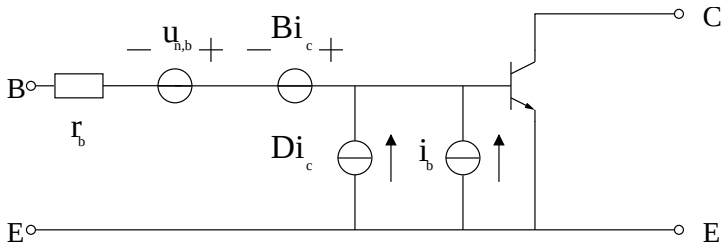

Figure 3.25: Noise sources placed at the input port.

Noise voltage and current sources are reduced separately because the equivalent u_n and i_n are substituted in the required equation. There is no need for further simplification since it is already done for the amplifier as a whole. For this configuration noise sources are:

$$u_n = 4kTr_b + \left(\frac{1}{g_m}\right)2qI_c \tag{3.35}$$

$$i_n = 2qI_b + \left(\frac{1}{\beta}\right)2qI_c \tag{3.36}$$

Just a quick reminder, spectral noise density calculation requires to square the value for any coefficient different of the noise source, i.e., $1/g_m$ and $1/\beta$ must be squared when perform calculations related to spectral density noise.

3.8.2 Common Base

The Chain-matrix for a BJT device in common base configuration is:

$$K_{CB} = \begin{pmatrix} \frac{1}{r_o g_m + 1} & \frac{r_o}{r_o g_m + 1} \\ \\ \frac{1}{r_o \beta + r_\pi} & \frac{\beta r_o + r_\pi + r_o}{r_o \beta + r_\pi} \end{pmatrix} \tag{3.37}$$

Figure 3.26 shows where noise sources are located.

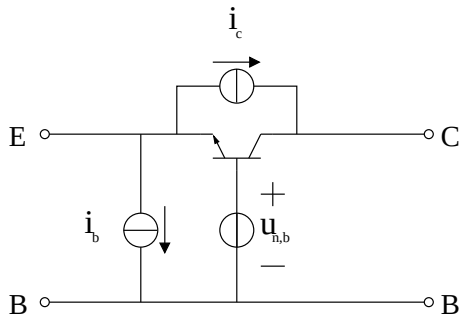

Figure 3.26: Noise sources for BJT in Common Base.

Unfortunately this configuration is not so easy to simplify, though. However the noise source movement can be performed in two steps.

Step 1.

V-shift is performed on $u_{n,b}$ and I-shift on i_c places sources at input and output ports of the device. Noise current source simplification can be done since i_b and i_c are in parallel. Figure 3.27

Step 2.

Two-port transformation to move $u_{n,b}$ and i_c located at the output port to the input port. Now all voltage sources are gathered in a single equivalent noise voltage source while current sources are gathered as well. Figure 3.28.

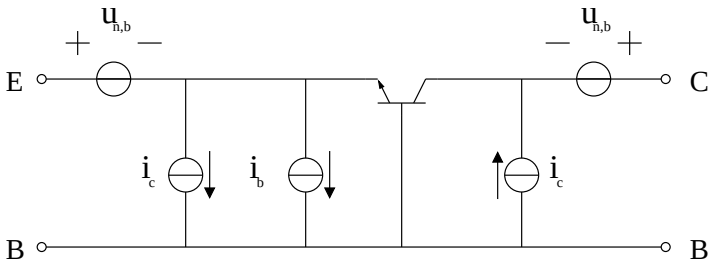

Figure 3.27: Noise sources movement.

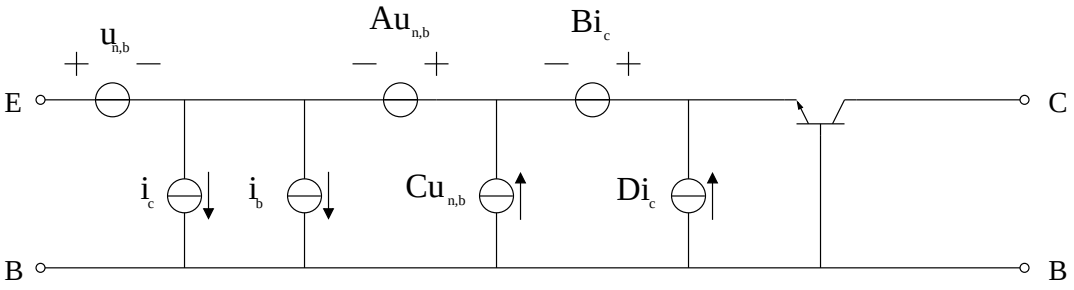

Figure 3.28: All noise sources are placed at the input port.

Equivalent noise sources are given by:

$$u_n = \left(\frac{g_m r_o}{1 + g_m r_o}\right) 4kTr_b + \left(\frac{r_o}{1 + g_m r_o}\right) 2qI_c \tag{3.38}$$

$$i_n = \left(\frac{1}{\beta r_o + r_\pi}\right) 4kTr_b + \left(\frac{r_o}{\beta r_o + r_\pi}\right) 2qI_c + 2qI_b \tag{3.39}$$

Remember to square coefficient values when performing spectral noise density values.

3.8.3 Common Collector

The Chain-matrix for a BJT device in common collector configuration is:

$$K_{CC} = \begin{pmatrix} \frac{\beta r_o + r_\pi + r_o}{\beta r_o + r_o} & \frac{r_\pi}{\beta + 1} \\ \\ \frac{1}{r_o \beta + r_o} & \frac{1}{\beta + 1} \end{pmatrix} \tag{3.40}$$

Noise sources are placed as shown in Figure 3.29.
Simplification can be done in two simple steps.

Step 1.

I-shift is applied to i_b. Figure 3.30.

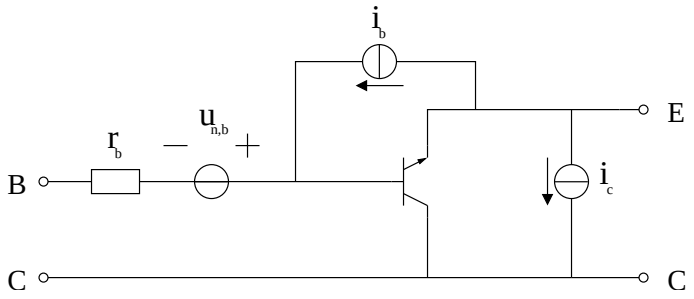

Figure 3.29: Noise sources for BJT in Common Collector.

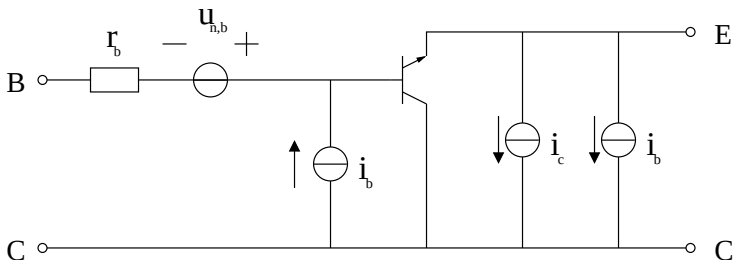

Figure 3.30: Noise source movement.

Step 2.

Two-port transformation on i_c and i_b moves these sources from the output port to the input port. Figure 3.31.

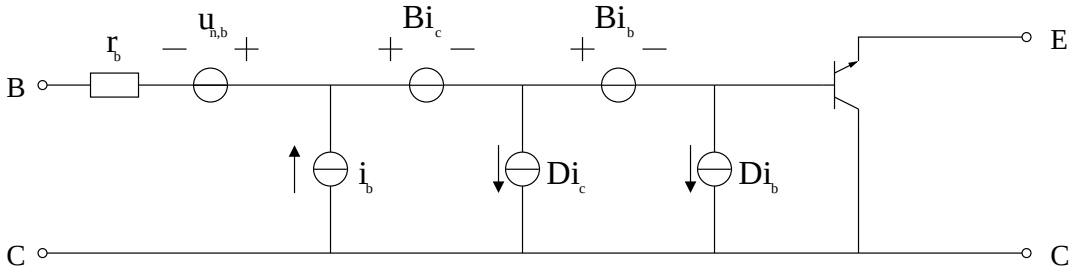

Figure 3.31: All noise sources are located at the input port.

The equivalent noise sources are:

$$u_n = 4kTr_b + \left(\frac{r_\pi}{\beta+1}\right)2qI_c + \left(\frac{r_\pi}{\beta+1}\right)2qI_b \tag{3.41}$$

$$i_n = \left(\frac{\beta}{\beta+1}\right)2qI_b + \left(\frac{1}{\beta+1}\right)2qI_c \tag{3.42}$$

Remember to square coefficient values when performing spectral noise density values.

3.8.4 BJT Differential Configurations

Single device configurations are easy to use but the structured design stablishes that in order to guarantee a negative-feedback behaviour it may be possible to employ differential configurations of BJT devices. At this point is not possible to stablish if this differential configuration should be implemented at this stage or on the other two stages (clipping and bandwidth) to be synthesised. Consequently the equivalent noise equations for the differential configurations are provided.

3.8.4.1 Differential Common Emitter

The Chain-matrix for a BJT device in common base configuration is:

$$K_{CE} = \begin{pmatrix} -\frac{1}{r_o g_m} & -\frac{2}{g_m} \\ -\frac{1}{2r_o\beta} & -\frac{1}{\beta} \end{pmatrix} \tag{3.43}$$

Figure 3.32 shows the differential configuration including its noise sources.

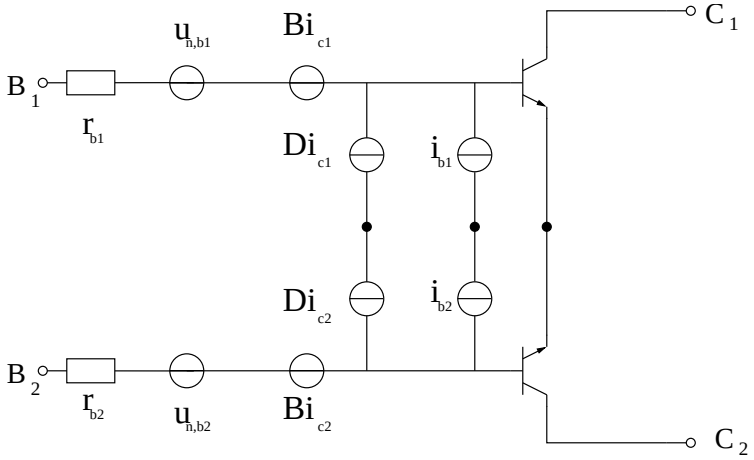

Figure 3.32: Differential common emitter with noise sources.

If both transistors are equal and biased at the same level, noise sources can be added. The result after reducing noise sources is a model that looks much as the single common emitter model. The difference is that equivalent noise voltage source is doubled and equivalent current source is reduced by half. Figure 3.33.

The noise characteristics for this configuration is similar to the single device but with subtle changes. Equivalent noise sources values are given by:

$$u_n = (2)\, 4kTr_b + \left(\frac{4}{g_m}\right) 2qIc \tag{3.44}$$

$$i_n = \left(\frac{1}{2}\right) 2qIb + \left(\frac{1}{2\beta}\right) 2qIc \tag{3.45}$$

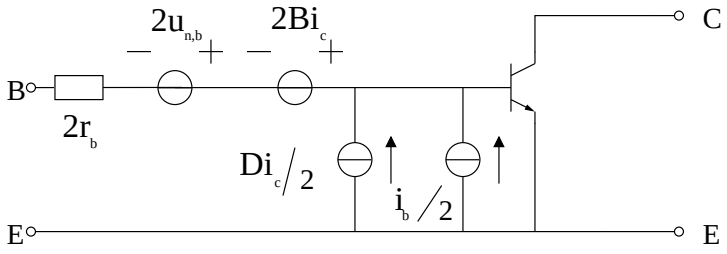

Figure 3.33: Simplified differential common emitter.

3.8.4.2 Differential Common Base

The Chain-matrix for a BJT device in differentail common base configuration is:

$$K_{CE} = \begin{pmatrix} \frac{1}{1+r_o g_m} & \frac{2r_o}{1+g_m r_o} \\ \frac{1}{2r_o\beta + r_\pi} & \frac{\beta r_o + r_o + r_\pi}{\beta r_o + r_\pi} \end{pmatrix} \tag{3.46}$$

Figure 3.34 shows the differential configuration including its noise sources.

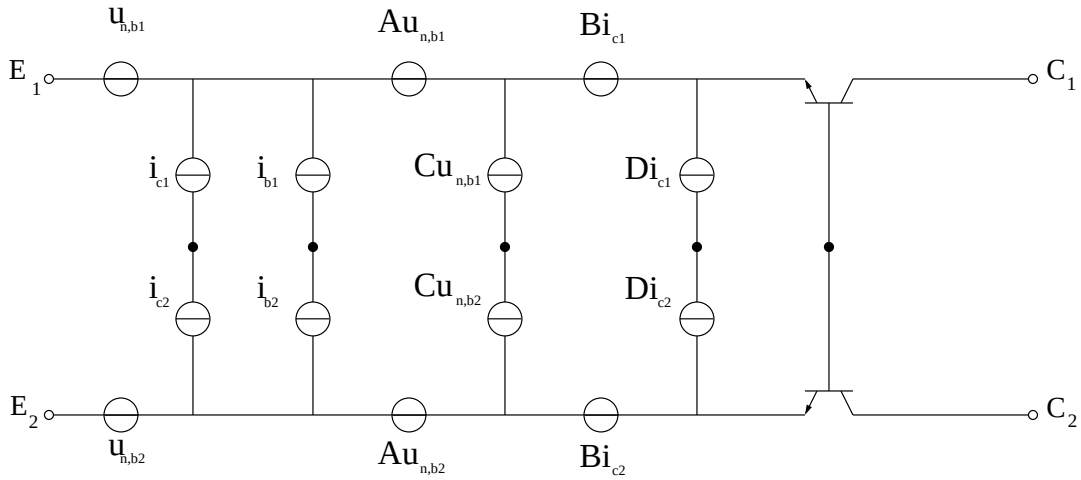

Figure 3.34: Differential common base with noise sources.

If both transistors are equal and biased at the same level, noise sources can be added. The result after reducing noise sources is a model that looks much as the single common base model. The difference is that equivalent noise voltage source is doubled and equivalent current source is reduced by half. Figure 3.35,

The noise characteristics for this configuration is similar to the single device but with subtle changes. Equivalent noise sources values are given by:

$$u_n = \left(\frac{2g_m r_o}{1+g_m r_o}\right) 4kTr_b + \left(\frac{4r_o}{1+g_m r_o}\right) 2qIc \tag{3.47}$$

$$i_n = \left(\frac{1}{4\beta r_o + 4r_\pi}\right) 4kTr_b + \left(\frac{r_o}{2\beta r_o + 2r_\pi}\right) 2qIc + \left(\frac{1}{2}\right) 2qIc \tag{3.48}$$

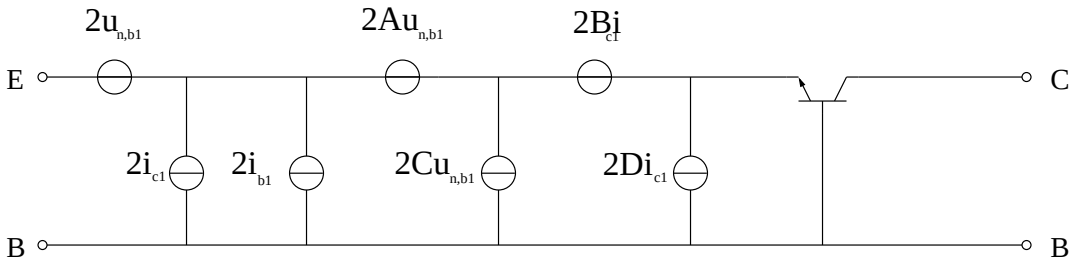

Figure 3.35: Simplified differential common base.

3.8.4.3 Differential Common Collector

The Chain-matrix for a BJT device in differentail common collector configuration is:

$$K_{CE} = \begin{pmatrix} \frac{\beta r_o + r_\pi + r_o}{\beta r_o + r_o} & \frac{2r_\pi}{\beta+1} \\ \\ \frac{1}{2\beta r_o + 2r_o} & \frac{1}{\beta+1} \end{pmatrix} \tag{3.49}$$

Figure 3.36 shows the differential configuration including its noise sources.

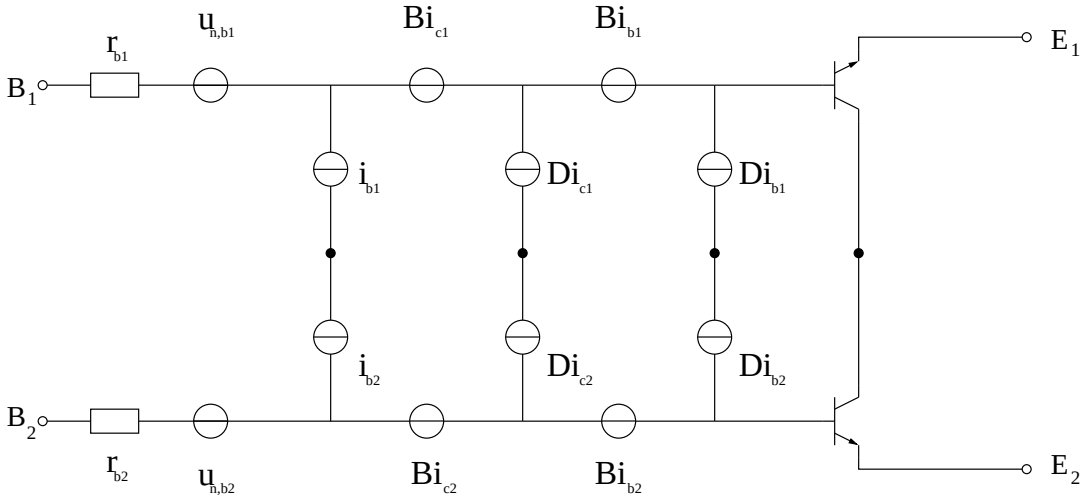

Figure 3.36: Differential common collector with noise sources.

If both transistors are equal and biased at the same level, noise sources can be added. The result after reducing noise sources is a model that looks much as the single common base model. The difference is that equivalent noise voltage source is doubled and equivalent current source is reduced by half. Figure 3.37.

The noise characteristics for this configuration is similar to the single device but with subtle changes. Equivalent noise sources values are given by:

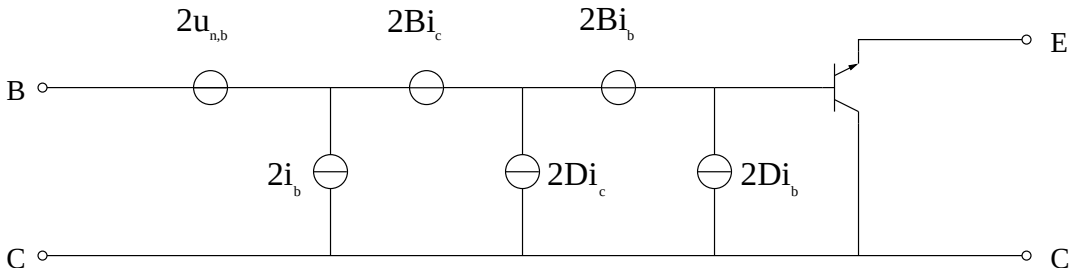

Figure 3.37: Simplified differential common collector.

$$u_n = (2)\,4kTr_b + \left(\frac{4r_\pi}{\beta+1}\right)2qIb + \left(\frac{4r_\pi}{\beta+1}\right)2qIc \tag{3.50}$$

$$i_n = \left(\frac{\beta}{2\beta+2}\right)2qIb + \left(\frac{1}{2\beta+2}\right)2qIc \tag{3.51}$$

3.9 Noise Design Algorithm

A flow diagram for the design of noise stage is shown in Figure 3.38.

The procedure is described as follows:

Input Basic Data

Process starts by requiring the input of basic data:

- Amplifier Type - This option could be: Voltage, Transadmittance, Transimpedance or Current Amplifier.
- Configuration - Configuration means that it could be either single-loop or double-loop (B).
- Source Impedance Type - Source impdance could be a resistor, capacitor or an inductance.
- Source Impedance Value - The related value to the impedance type (ohms, farads or henrys).

Input Noise Values

Once basic data has been provided, it is necessary to specify the desired noise level to be accomplished by the amplifier. Noise values to be required ad:

- Noise Unit - Could be either decibels (dB) or spectral noise density given in terms of either V^2/Hz or A^2/Hz.
- Noise Value - The numerical noise value to be achieved.

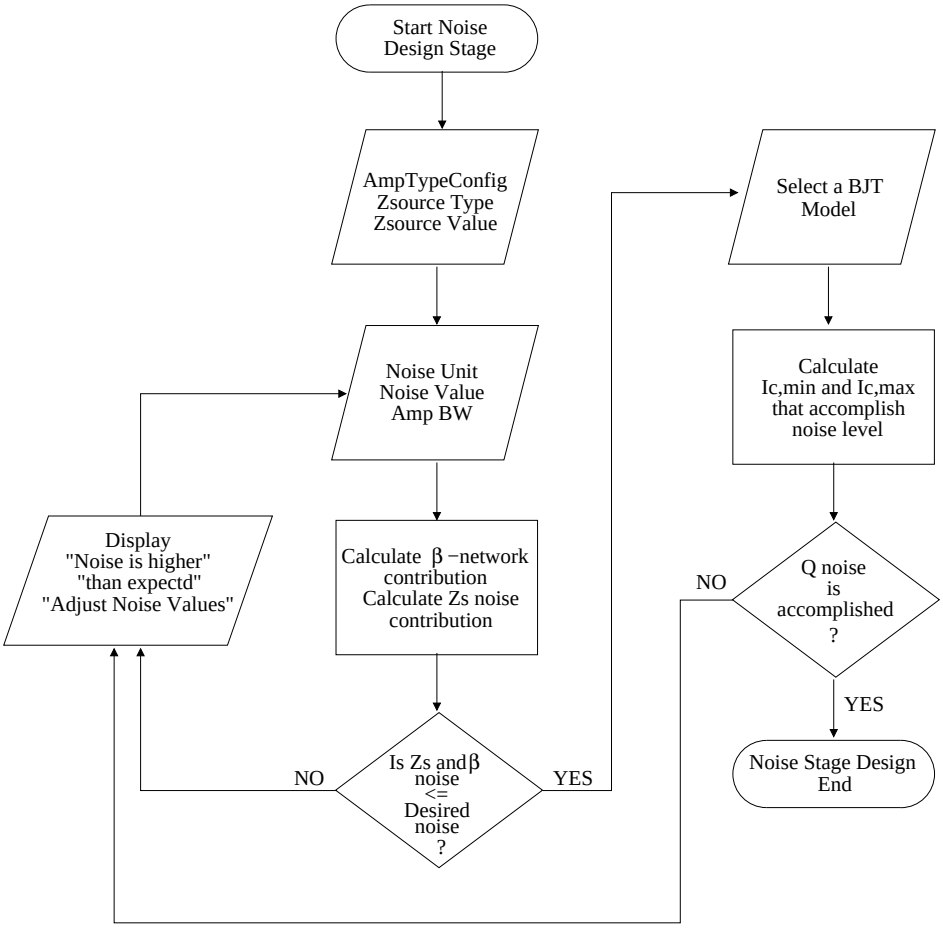

Figure 3.38: Flow diagram for noise stage design.

- Bandwidth Value - The bandwidth value that the amplifier is going to handle. This value is useful when the source impedance is a capacitor or an inductor since both are frequency dependent devices.

Calculate Noise Contributions

Given that values for source and feedback network are already provided it is possible to calculate their noise contributions.

Compare Noise Values

The sum of noise from the feedback network and source impedance must be smaller than the noise value provided. In case that it is greater, a message must be displayed indicating that noise value is higher than expected. Therefore the noise parameters should be adjusted.

Input Active Device

Noise values for feedback and source are known and are below the noise value established as limit. There is still some noise "room" to operate below the noise limit for the design. Selecting a model number (i.e. BC555, 2N2222) provides some useful parameters (r_b, β) to perform noise calculation for the equivalent noise sources i_n and u_n.

Calculate Ic Values

Specific values for Ic provide specific values of noise since parameters as $r_p i$, r_o and g_m depends on it. It is necessary to calculate which Ic values provides equivalent noise sources that performs below the noise limit. There is going to be an $I_{c,min}$ and an $I_{c,max}$ values. These values must be kept for further operations.

Compare Q Noise Values

All noise source values are calculated and must be added. The resulting value must be smaller or equal at least to the proposed noise level. If this value is greater, a message is displayed indicating the anomaly and noise parameters should be adjusted. Otherwise the noise stage design is terminated and nullor synthesis process continues.

Chapter 4

Distortion Stage

4.1 Introduction

Semiconductor devices are inherently nonlinear. For BJT transistors that operates in forward active region, exhibit an exponential relationship between the collector current and the base-emitter voltage. For MOS transistors in saturated region, the relationship between the drain current and the gate-source voltage obeys a square law. Circuits made up with active components exhibit certain amount of nonlinearity, this means that the relationship between their input and the output variables is not so linear as one might expect.

Active devices that are employed in analog signal processing applications are operated in a quasi-linear region [58]. The linearity assumption is valid especially when signals with small amplitude are processed. Nevertheless, designers are requested to evaluate the limits of the linear approximation or to characterise the effects of nonlinear distortion in circuits and systems assumed as linear blocks [59].

For the structured design concern, the design of the distortion stage stablishes that this block should be located at the output of the synthesised nullor (Figure 4.1). Its non-linear characteristic of the active device employed on the nullor synthesis, may cause certain contribution to the transferred signal.

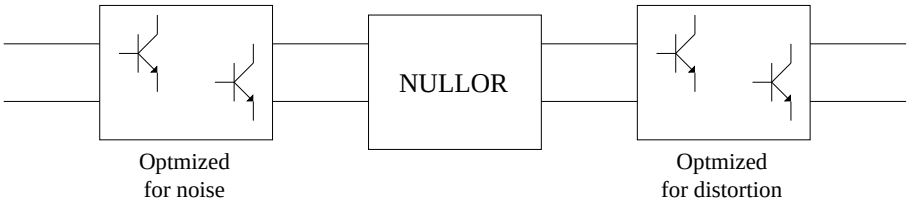

Figure 4.1: Distortion stage design placement.

This contribution is reduced by a proper choice for the topology of this last stage and a correct, though adequate, correct biasing of the stage. These levels should be sufficiently great to avoid *clipping* of the amplifier at the maximally occurring signal levels for currents and their associated voltages. *Clipping* is not just the only distortion that can be avoided, *weak nonlinear distortion* remains as a contribution of the active circuit output stage.

4.2 Distortion Types

As explained on the above lines, the nonlinearities in the transfer function of an active device (such as vacuum tubes, transistors and operational amplifiers) are a common source of non-linear distortion; in passive components (such as a coaxial cable or optical fiber), linear distortion can be caused by inhomogeneties, reflections, and so on in the propagation path. The are other kinds of distortion but are outside the scope of this work. For electronic signals distortion is classified as follows [60]:

- Amplitude distortion - Is distortion ocurring in a system, subsystem, or device when the output amplitude is not a *linear function* of the input amplitude under specific conditions.
- Crossover distortion - This kind of distortion appears in *class-B push-pull* amplifiers where, due to the forward voltage of the *Base-Emitter* junction of the output BJTs, the output signal does not follow the input until the input exceeds this forward voltage.
- Frequency distortion - This form of distortion occurs when different frequencies are amplified by different amounts, mainly caused by combination of active device and components. For example, the non-uniform frequency response curve of an RC-coupled cascade amplifier.
- Phase distortion - Mostly occurs due to a reactive component, such as capacitive reactance or inductor capacitance. Here, all the components of the input signal are not amplified with the same phase shift, hence causing some parts of the output signal to be out of phase with the rest of the output.

Non-linear distortions considered for the structured design methodology are [1]:

- Clipping distortion.
- Weak distortion.

4.2.1 Clipping Distortion

Is a consequence of the fact that a nonlinear device is biased. This bias level may appear to be insufficient, resulting in clipping distortion. Bipolar transistors should operate in the forward active region. The collector-emitter voltage should always be larger than the saturation voltage. For lower voltages, the bipolar transistors starts saturating and in consequence the gain is reduced. On top of that, for proper functioning, the collector should be positive [1].

4.2.2 Weak Distortion

Weak distortion is caused by the weak nonlinearity of the input-output relation of the device. When considering weak distortion it is assumed that the proper bias signals are added, i.e., no clipping distortion present and the required small-signal behavior is obtained. For bipolar transistors there are two types of weak distortion [1]:

- β-distortion.
- g_m-distortion.

The first type of distortion arises in the bipolar transistor when it is current driven. For the structured design this is a consequence of the assumption that the current-gain factor of the bipolar transistor is independent of the collector current, i.e., a constant. For the g_m distortion, this is caused by the collector-current dependency of the transconductance of the transistors. For instance, when the driving resistance is ideal (that means, zero), an ideal voltage drive and g_m distortion dominates in the transistor.

4.3 Clipping Distortion Considerations

Clipping occurs when signal becomes so large that the small-signal model is not valid anymore. It is likely that this type of distortion occurs in the last stage since this stage has to handle the largest signal (voltage and current). This stage has to be biased at a current and a voltage large enough for delivering the necessary current and voltage to the load in order to mantain the desired gain level. Figure 4.2 shows how the amplifier is designed so far.

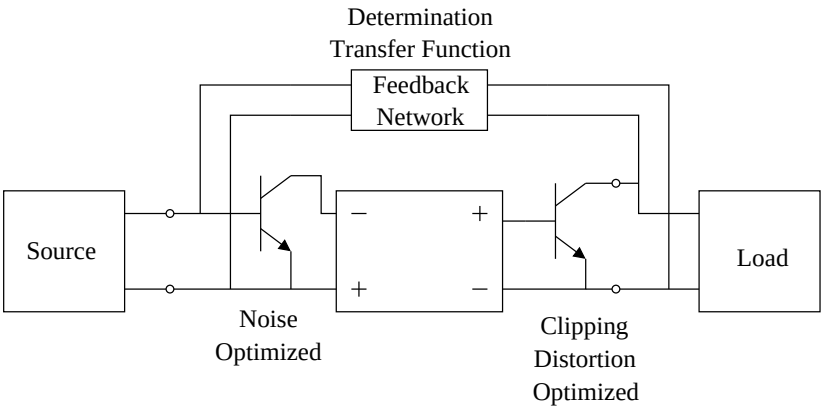

Figure 4.2: Amplifier design with first and last stages designed.

It is worth to note that the two-port block placed between the two transistors is employed to represent intermediate stages and is not a nullor.

One important thing to beware of by the designer is the maximum voltage and current that the amplifier has to handle. Both the load of the circuit and the feedback network require power that has to be delivered by the output stage. It is undesirable to operate the transistor at its limits, as it is already explained.

4.3.1 Clipping in Intermediate Stages

If it is necessary to implement an intermediate stage, the designer must assure that this stage is able to drive the entire frequency range over the last stage. Care should be taken when a capacitor is employed for coupling stages because if the bias for the intermediate stage is too low, the *entire* bias current for this stage is used to charge the input capacitance

for the next stage. This situation makes that intermediate stage is switched off breaking the negative-feedback loop. When conditions settle down and the stage switches on, the entire amplifier will take some time to normalise its operation. The time it takes to settle might be longer than the input frequency and this situation creates transients at the output of the amplifier. Keep in mind that the minimal bias current for the intermediate stage is given by the maximal input current of the next stage.

4.4 Equations

4.4.1 Voltage Amplifier

The topology for this amplifier can be seen in Figure 4.3 [60].

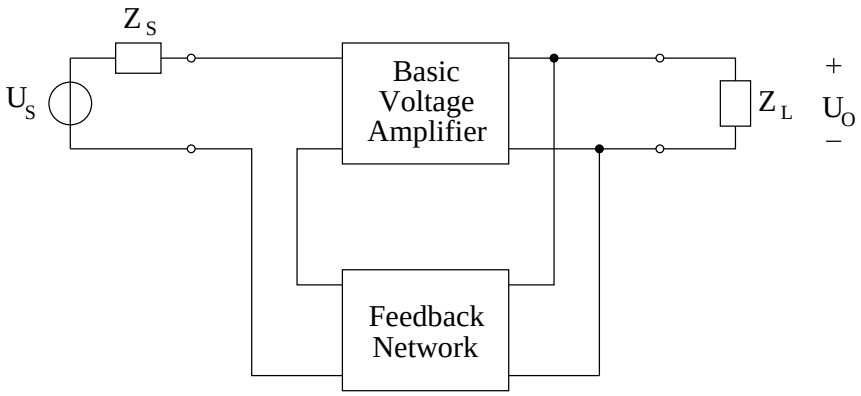

Figure 4.3: Series-Parallel topology.

Essentially this type of amplifier is a voltage controlled voltage source. The feedback topology for this amplifier is known as *series-parallel feedback*, where *series* is related to the input connection while *parallel* refers to the output connection.

To avoid clipping it is necessary to know these values:

- Equivalent output impedance load.
- Maximum current output.
- Maximum voltage output.

Data is calculated in five simple steps:

Step 1 Find out the peak value for the output signal, in this case the output signal is voltage. This is known by multiplying input level value by the desired gain value, the result is then multiplied by $\sqrt{2}$.

$$V_{out,max} = V_{in} \times Gain \times \sqrt{2} \tag{4.1}$$

Step 2 The input of the amplifier is at ground level because nullor properties, therefore the maximum load conditions are given by load impedance in parallel with the feedback network (R_1 *and* R_2 *are in series*). Therefore:

$$Z_{load,output} = \frac{Z_L Z_{feed}}{Z_L + Z_{feed}} \tag{4.2}$$

where

$$Z_{feed} = R_1 + R_2 \tag{4.3}$$

Step 3 By applying Ohm's Law [61] the maximum output current is given by:

$$I_{out,max} = \frac{V_{out,max}}{Z_{load,output}} \tag{4.4}$$

Step 4 In order to find the minimum collector-emitter bias voltage value it is necessary to stablish that base-collector voltage is negative for NPN devices, or for PNP devices base-collector voltage is positive. Then for NPN devices:

$$V_{CE,min} = V_{out,max} + V_{be} = V_{out,max} + 0.7 \tag{4.5}$$

Step 5 Last value is the maximum bias current, in terms of collector current. The value is chosen 50% beyond the minimum to avoid a critically biased device. Thus,

$$I_C = 1.5 \times I_{out,max} \tag{4.6}$$

4.4.2 Transadmittance Amplifier

The topology for this amplifier can be seen in Figure 4.4 [60].

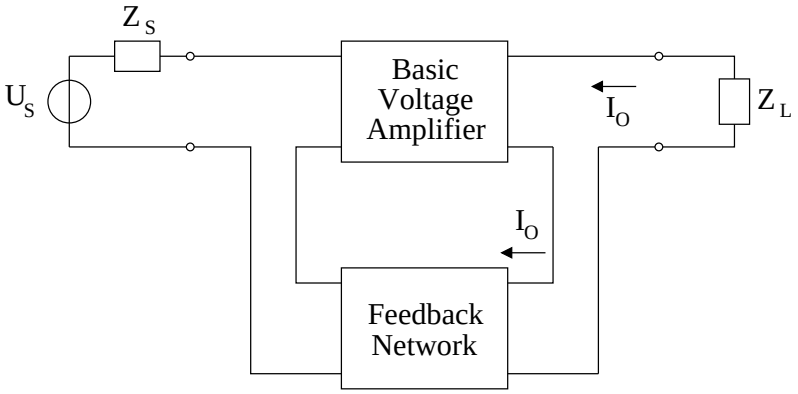

Figure 4.4: Series-Series topology.

The feedback topology for this amplifier is known as *series-series feedback*, where *series* is related to the input connection and to the output connection.

As with the previous amplifier, these values are necessary to be known:

- Equivalent output impedance load.
- Maximum current output.
- Maximum voltage output.

Only five steps are required:

Step 1 Find out the peak value for the output signal, in this case the output signal is current. Found by multiplying input level value by the desired gain value, the result is then multiplied by $\sqrt{2}$.

$$I_{out,max} = V_{in} \times Gain \times \sqrt{2} \tag{4.7}$$

Step 2 The input of the amplifier is at ground level because nullor properties, therefore the maximum load conditions are given by load impedance in series with the feedback network (R_1). Therefore:

$$Z_{load,output} = Z_L + Z_{feed} \tag{4.8}$$

where

$$Z_{feed} = R_1 \tag{4.9}$$

Step 3 By applying Ohm's Law [61] the maximum output voltage is given by:

$$V_{out,max} = I_{out,max} \times Z_{load,output} \tag{4.10}$$

Step 4 In order to find the minimum collector-emitter bias voltage value it is necessary to stablish that base-collector voltage is negative for NPN devices, or for PNP devices base-collector voltage is positive. Then for NPN devices:

$$V_{CE,min} = V_{out,max} + V_{be} = V_{out,max} + 0.7 \tag{4.11}$$

Step 5 Last value is the maximum bias current, in terms of collector current. The value is chosen 50% beyond the minimum to avoid a critically biased device. Thus,

$$I_C = 1.5 \times I_{out,max} \tag{4.12}$$

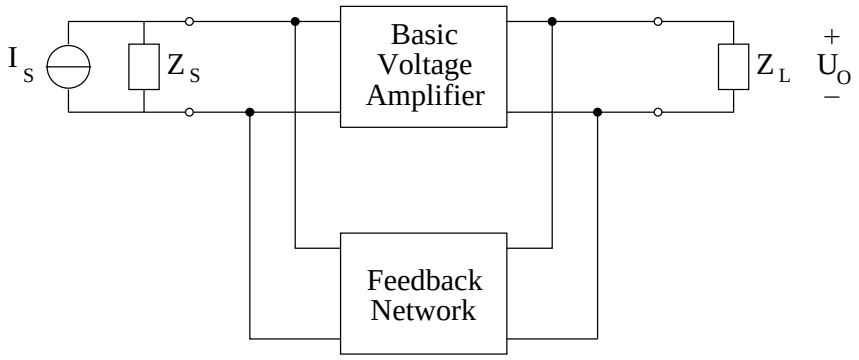

Figure 4.5: Parallel-Parallel topology.

4.4.3 Transimpedance Amplifier

The topology for this amplifier can be seen in Figure 4.5 [60].

The feedback topology for this amplifier is known as *parallel-parallel feedback*, where *parallel* is related to the input connection and to the output connection.

Values needed to be known:

- Equivalent output impedance load.
- Maximum current output.
- Maximum voltage output.

Again, the five steps approach:

Step 1 Find out the peak value for the output signal, in this case the output signal is voltage. Found by multiplying input level value by the desired gain value, the result is then multiplied by $\sqrt{2}$.

$$V_{out,max} = I_{in} \times Gain \times \sqrt{2} \tag{4.13}$$

Step 2 The input of the amplifier is at ground level because nullor properties, therefore the maximum load conditions are given by load impedance in parallel to the feedback network (R_1). Therefore:

$$Z_{load,output} = \frac{Z_L Z_{feed}}{Z_L + Z_{feed}} \tag{4.14}$$

where

$$Z_{feed} = R_1 \tag{4.15}$$

Step 3 By applying Ohm's Law [61] the maximum output current is given by:

$$I_{out,max} = \frac{V_{out,max}}{Z_{load,output}} \tag{4.16}$$

Step 4 In order to find the minimum collector-emitter bias voltage value it is necessary to stablish that base-collector voltage is negative for NPN devices, or for PNP devices base-collector voltage is positive. Then for NPN devices:

$$V_{CE,min} = V_{out,max} + V_{be} = V_{out,max} + 0.7 \tag{4.17}$$

Step 5 Last value is the maximum bias current, in terms of collector current. The value is chosen 50% beyond the minimum to avoid a critically biased device. Thus,

$$I_C = 1.5 \times I_{out,max} \tag{4.18}$$

4.4.4 Current Amplifier

The topology for this amplifier can be seen in Figure 4.6 [60].

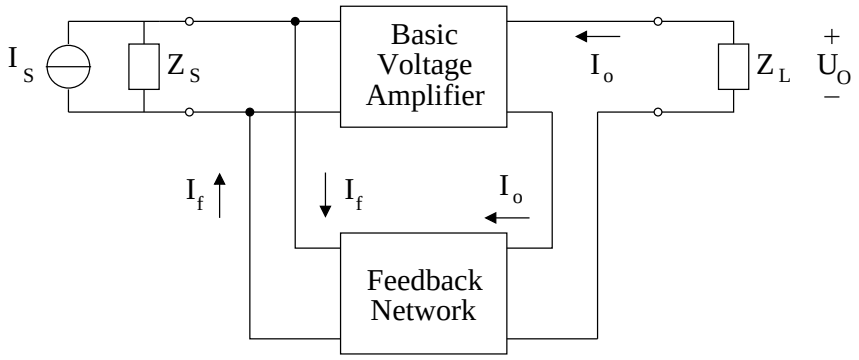

Figure 4.6: Parallel-Series topology.

The feedback topology for this amplifier is known as *parallel-series feedback*, where *parallel* is related to the input connection and *series* to the output connection.

Important values to be known:

- Equivalent output impedance load.
- Maximum current output.
- Maximum voltage output.

This is the five steps approach:

Step 1 Find out the peak value for the output signal, in this case the output signal is current. Found by multiplying input level value by the desired gain value, the result is then multiplied by $\sqrt{2}$.

$$I_{out,max} = I_{in} \times Gain \times \sqrt{2} \tag{4.19}$$

Step 2 The input of the amplifier is at ground level because nullor properties, therefore the maximum load conditions are given by load impedance in series to the feedback network (R_1 *and* R_2 *are in parallel*). Therefore:

$$Z_{load,output} = Z_L + Z_{feed} \tag{4.20}$$

where

$$Z_{feed} = \frac{R_1 R_2}{R_1 + R_2} \tag{4.21}$$

Step 3 By applying Ohm's Law [61] the maximum output voltage is given by:

$$V_{out,max} = I_{out,max} \times Z_{load,output} \tag{4.22}$$

Step 4 In order to find the minimum collector-emitter bias voltage value it is necessary to stablish that base-collector voltage is negative for NPN devices, or for PNP devices base-collector voltage is positive. Then for NPN devices:

$$V_{CE,min} = V_{out,max} + V_{be} = V_{out,max} + 0.7 \tag{4.23}$$

Step 5 Last value is the maximum bias current, in terms of collector current. The value is chosen 50% beyond the minimum to avoid a critically biased device. Thus,

$$I_C = 1.5 \times I_{out,max} \tag{4.24}$$

4.4.5 Two-loop (B)

This kind of topology allows to have all four kinds of transfers (voltage, transadmittance, transimpedance and current) without any change of components, just apply the required source and place an adequate load. As for the clipping distortion stage, the basic value to be known is the output impedance. It is possible to obtain the output impedance using the Chain-matrix results [61], that is, transform Chain-matrix to an impedance matrix. This can be done using Table 4.1, note that here Chain-matrix parameters are represented as $\mathbf{T}_{xx}$ and $\Delta_{\mathbf{T}} = \mathbf{T}_{11}\mathbf{T}_{22} - \mathbf{T}_{12}\mathbf{T}_{21}$.

For one-loop topologies the transformation cannot be performed since all parameters but one are equal to zero because of nullor characteristics. The parameters given by Chain-matrix for two-loop amplifiers are different from zero. Output impedance for nullor is found by

	T
Z	$\frac{\mathbf{T}_{11}}{\mathbf{T}_{21}}$ $\frac{\Delta_{\mathbf{T}}}{\mathbf{T}_{21}}$ $\frac{1}{\mathbf{T}_{21}}$ $\frac{\mathbf{T}_{22}}{\mathbf{T}_{21}}$

Table 4.1: Chain-matrix to impedance matrix transformation.

$$Z_{out_nullor} = \frac{Z_{11}Z_{22} - Z_{12}Z_{21}}{Z_{11}} \tag{4.25}$$

Notice that no matter what type of amplified is designed, output impedance is the same for all four possibilities. Design steps can be separated in two cases shown in Figure 4.7 and Figure 4.8.

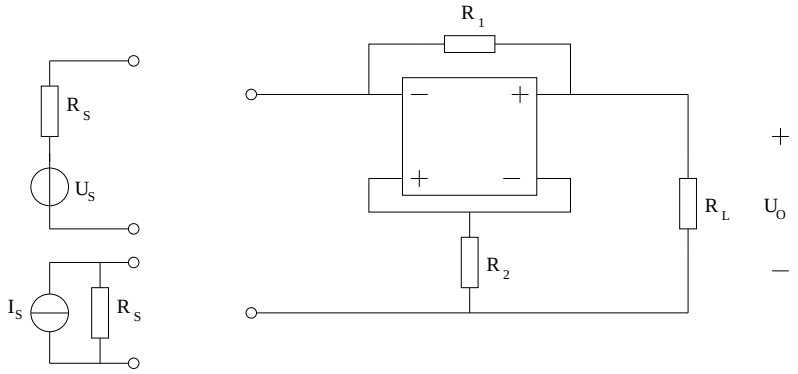

Figure 4.7: Two-loop (B) voltage output.

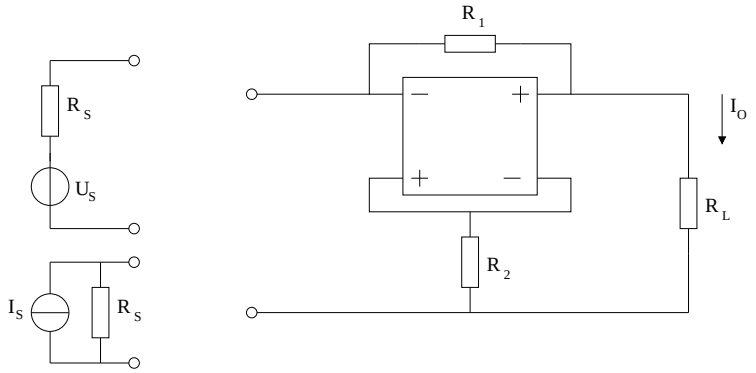

Figure 4.8: Two-loop (B) current output.

For both cases the equivalent output load impedance is given by the parallel of R_L and Z_{out_nullor}, therefore

$$Z_{load,output} = \frac{Z_{out_nullor} R_L}{Z_{out_nullor} + R_L} \tag{4.26}$$

Voltage Output Two-loop Amplifier

Step 1 Output signal is voltage, then peak value is found according to these two cases:

1. **Input Voltage**.
$$V_{out,max} = V_{in} \times \left(\frac{-R_2}{R_1 - R_2} \right) \times \sqrt{2} \tag{4.27}$$
The result is a negative voltage, it means that output signal phase is inverted.
2. **Input Current**
$$V_{out,max} = I_{in} \times \left(-\frac{R_1 R_2}{R_1 - R_2} \right) \times \sqrt{2} \tag{4.28}$$
Output signal is inverted, that is the reason for the negative value.

Step 2 By applying Ohm's Law [61] the maximum output current is given by:

$$I_{out,max} = \frac{V_{out,max}}{Z_{load,output}} \tag{4.29}$$

Step 3 In order to find the minimum collector-emitter bias voltage value it is necessary to stablish that base-collector voltage is negative for NPN devices, or for PNP devices base-collector voltage is positive. Then for NPN devices:

$$V_{CE,min} = V_{out,max} + V_{be} = V_{out,max} + 0.7 \tag{4.30}$$

Step 4 Last, the maximum bias current is calculated in terms of collector current. The value is chosen 50% beyond the minimum to avoid a critically biased device. Thus,

$$I_C = 1.5 \times I_{out,max} \tag{4.31}$$

Current Output Two-loop Amplifier

Step 1 The output signal is current and two cases apply. Gain values are negative because phase of input signal is inverted.

1. **Input Voltage**.
$$I_{out,max} = \frac{V_{in}}{\left(-\frac{1}{R_1 - R_2} \right)} \times \sqrt{2} \tag{4.32}$$
2. **Input Current**
$$I_{out,max} = I_{in} \times \left(-\frac{R_2}{R_1 - R_2} \right) \times \sqrt{2} \tag{4.33}$$

Step 2 By applying Ohm's Law [61] the maximum output voltage is given by:

$$V_{out,max} = I_{out,max} \times Z_{load,output} \tag{4.34}$$

Step 3 In order to find the minimum collector-emitter bias voltage value it is necessary to stablish that base-collector voltage is negative for NPN devices, or for PNP devices base-collector voltage is positive. Then for NPN devices:

$$V_{CE,min} = V_{out,max} + V_{be} = V_{out,max} + 0.7 \tag{4.35}$$

Step 4 Last value is the maximum bias current, in terms of collector current. The value is chosen 50% beyond the minimum to avoid a critically biased device. Thus,

$$I_C = 1.5 \times I_{out,max} \tag{4.36}$$

4.5 Clipping Design Algorithm

The flow diagram for this stage is shown in Figure 4.9.

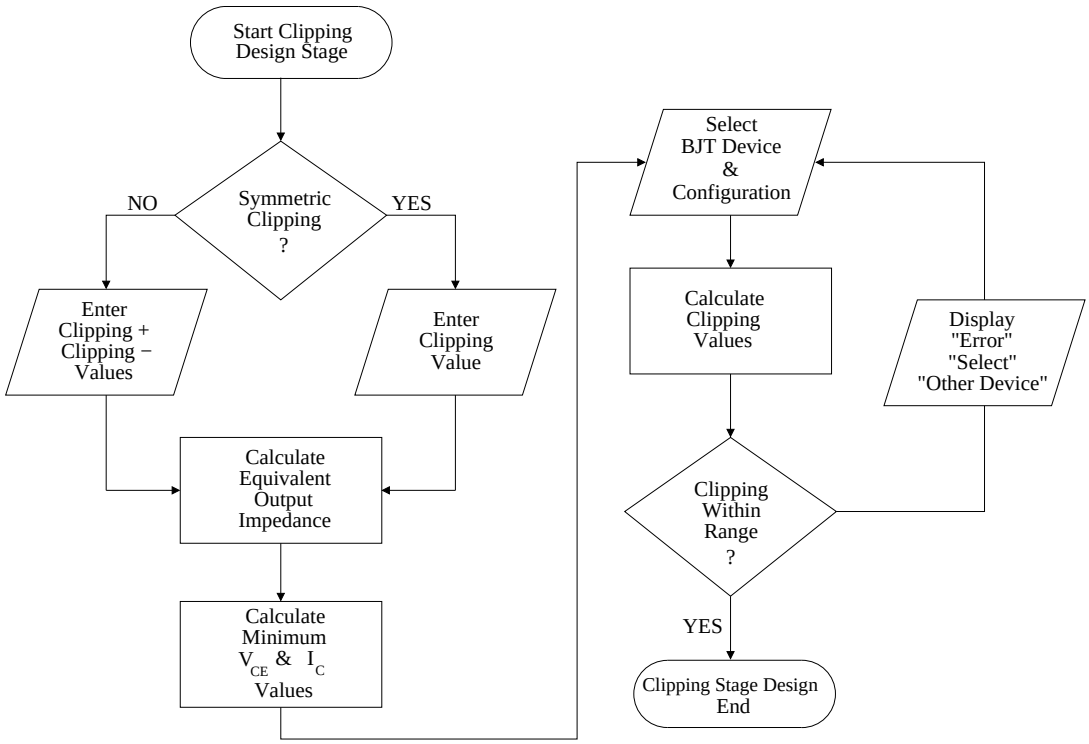

Figure 4.9: Flow diagram for clipping stage design.

Procedure is as follows:

Input Clipping Values

There is an option for the designer to decide if clipping value is symmetrical or individual values are provided. This value is the maximum allowable clipping for the design.

Equivalent Output Impedance

Output impedance is calculated based on the amplifier type and based on the previous explanation.

Minimum Values for V_{CE} and I_C

Using the previous equations given for every configuration, minimum required values for V_{CE} and I_C are calculated.

BJT Selection

Here a BJT model must be selected and based on this model the maximum values provided on its data sheet are compared to the minimum required values already calculated.

Clipping Consideration

If clipping is confirmed, the clipping value (or values if it is not symmetrical) is (or are) compared to the value provided in the maximum allowable value. In case that clipping value is higher than the allowed, an error message must be issued and another device ought to be selected. Otherwise the process is finished and this stage is completely designed within constraints.

Chapter 5

Bandwidth Stage

5.1 Introduction

The design process has implemented, so far, the input and output stages for the nullor to be synthesized. These implementations addresses the noise and clipping considerations for the overall amplifier behaviour. The last consideration to be taken into account is related with the frequency domain behaviour of the amplifier.

On first hand, lower frequencies affect the coupling and bias capacitors since they no longer can be replaced by the short-circuit approximation because their reactances are increased [62]. On the other hand, the frequency dependent parameters employed in the models of the active devices and their associated parasitic capacitances, impose a limit for the high frequency behaviour of the amplifier. Consequently, an increase on the number of cascade stages for a system will also limit the response for higher and lower frequencies.

The design for the frequency behaviour of the amplifier can take a lot of time and work. Thus a need for a simple measure for the bandwidth capability of the amplifier helps to avoid the waste of time performing length and tedious calculations, to find out that the design will not be able to perform within the desired bandwidth. The structured design methodology provides a simple method to calculate the bandwidth capabilities of the design and, in case this value is not adequate, a process for adjustment.

Once the bandwidth capabilities for the amplifier are guaranteed, an additional step is necessary to complete the design of this stage. This step is aimed to obtain a flat transfer function over the frequency bandwidth. Such a flat transfer function is provided by an appropriate placement of the closed-loop system poles [1]. At this point, the system poles positions are related to the small-signal components values provided by the bias values for each designed stage. The orthogonal way to perform the design process makes impossible for the designer to know how the bias values for the preceding stages will affect the overall frequency behaviour, hence the need of an adequate placement of the system poles. It is well known that a flat response along the frequencies of the bandwidth range can be obtained by the *Butterworth approximation* [60], therefore it may be possible to apply concepts of the network synthesis to the amplifier design.

5.2 Asymptotic-gain Model

The best model for describing feedback systems is the *Black's model* [63], Figure 5.1. Methods based on this model are more suitable for analysis rather than for a simplified design method. The transfer function (A_t) for the model is given by:

$$A_t = \frac{E_l}{E_s} = \frac{A}{1 - Ak} \tag{5.1}$$

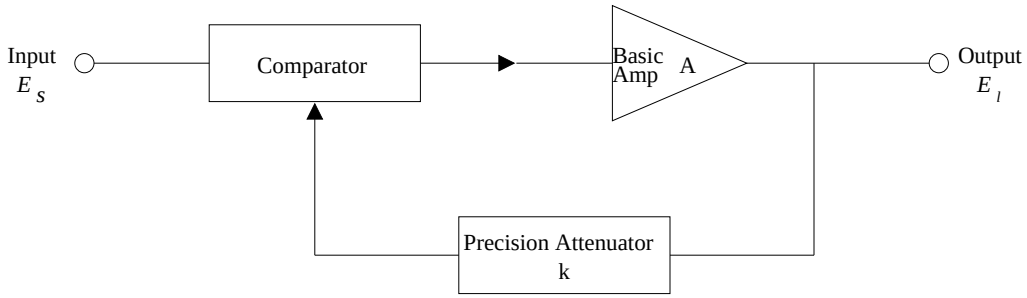

Figure 5.1: Black's feedback model.

Given the fact that this model has drawbacks because assumes ideal unilateral transfers and no loading effects at input and output of the system, a better model to describe the electronic feedback systems is the *asymptotic-gain model* [34]. This model is based on the superposition model shown in Figure 2.6.

As already shown, the transfer function for the superposition model is given as follows:

$$A_t = \frac{E_l}{E_s} = A_{t0} + \upsilon\xi\frac{A}{1 - A\beta} \tag{5.2}$$

where $A\beta$ is known as *loop-gain*, υ represents a non-ideal coupling between the input source and the control input of the controlled source, ξ represents a non-ideal coupling between the controlled source and the output, and A_{t0} is the direct transfer factor. Usually υ and ξ are equal to unity but if an incorrect amplifier type is chosen then they will have a different value, in fact ν and ξ are closely related with the issue of the impedance matching.

The higher the loop gain, more dominant the feedback β becomes on the overall transfer. The asymptotic gain ($A_{t\infty}$) is found when the loop-gain approaches infinity, that is

$$\lim_{A\beta\to\infty} A_t = A_{t\infty} = A_{t0} - \frac{\upsilon\xi}{\beta} \tag{5.3}$$

If $\upsilon = 1$ and $\xi = 1$, then the asymptotic gain becomes

$$A_{t\infty} = A_{t0} - \frac{1}{\beta} \tag{5.4}$$

In this way, the transfer function is expressed as

$$A_t = \frac{E_l}{E_s} = A_{t\infty}\frac{-A\beta}{1 - A\beta} + \frac{A_{t0}}{1 - A\beta} \tag{5.5}$$

For an infinite loop-gain the second term disappears and even in practical amplifiers the first term tends to dominate the second one [1], the transfer function of a negative-feedback amplifier can then be written as

$$A_t = A_{t\infty}\frac{-A\beta}{1-A\beta} \tag{5.6}$$

So for the case when there is an infinite loop-gain — like the nullor case — the transfer of the amplifier is

$$A_t = -\frac{1}{\beta} \tag{5.7}$$

an expression that can be implemented using just passive components since β is an attenuation.

5.3 The LP-product

At this stage of the nullor synthesis the need to know if the design can accomplish the frequency constraints becomes a priority. The structured design methodology provides a quick and efficient way to know if the amplifier can reach the desired bandwidth, it is known as Loop-gain-Poles product (*LP-product*) [34, 1]. It stands for a simple way of measuring the bandwidth capability of the nullor based amplifiers.

The LP-product calculation is derived from the asymptotic-gain model already explained in the previous section. Now if we assume that the gain depends on the frequency s, then the scheme is shown in Figure 5.2.

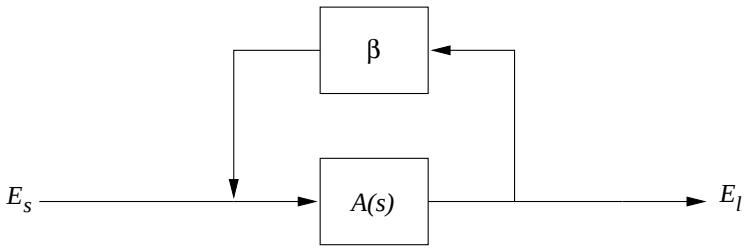

Figure 5.2: Basic scheme for the LP-product calculation.

Now the transfer function becomes

$$A_t = A_{t\infty}\frac{-A(s)\beta}{1-A(s)\beta} \tag{5.8}$$

recalling that

$$A_{t\infty} = A_{t0} - \frac{1}{\beta} \tag{5.9}$$

where the term A_{t0} can be neglected, therefore the transfer function is

$$A_t = -\frac{1}{\beta}\frac{-A(s)\beta}{1-A(s)\beta} \tag{5.10}$$

The LP-product calculation can be derived taking into account the poles from Eq. 5.10 and some assumptions. For instance, the denominator of a second-order transfer function is given by

$$\begin{aligned} d^{(2)}(s) &= (s+p_1)(s+p_2) \\ &= s^2 + (p_1+p_2)s + p_1p_2 \end{aligned} \tag{5.11}$$

For a third order system , it yields

$$\begin{aligned} d^{(3)}(s) &= (s+p_1)(s+p_2)(s+p_3) \\ &= s^3 + s^2(p_1+p_2+p_3) + s(p_1p_2+p_1p_3+p_2p_3) + p_1p_2p_3 \end{aligned} \tag{5.12}$$

It clearly results that for an n-order system, the denominator becomes

$$d^{(n)}(s) = s^n + s^{n-1}\sum_{i=1}^{n} p_i + \ldots + \prod_{i=1}^{n} p_i \tag{5.13}$$

That is to say:

- The independent coefficient equals the product of poles.
- The $(n-1)$ coefficient equals the sum of poles

It is possible to express the loop-gain in function of the DC-gain and the poles of the system. For the n-order system is expressed as

$$A(s)\beta = \frac{A(0)\beta}{(1-\frac{s}{p_1})\ldots(1-\frac{s}{p_n})} \tag{5.14}$$

By means of Eq. 5.14 and Eq. 5.10 is possible to obtain the characteristic polynomial (CP) of the total transfer $A_t(s)$, again for the n-order system it is found to be

$$CP(s) = s^n + s^{n-1}\sum_{i=1}^{n} p_i + \ldots + [1-A(0)\beta]\prod_{i=1}^{n} p_i \tag{5.15}$$

Butterworth poles placement is desired, that is, poles are placed on a half circle in the negative real axis. The radius of the circle equals the frequency of the transfer function. Figure 5.3 illustrates this subject for a second-order system.

The r from the figure refers to the radius of the half-circle, the subscript 2 indicates the order of the system. Since the Butterworth poles placement are already known, the characteristic polynomial is given by

$$CP(s) = s^n + s^{n-1}\sum^{b} n_{i=b1}p_i + \ldots + \prod^{b} n_{i=b1}p_i \tag{5.16}$$

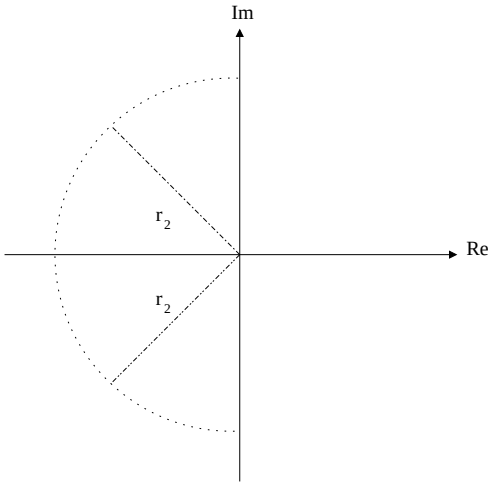

Figure 5.3: Butterworth poles placement for a second-order system.

here Butterworth poles are denoted as bx.

Comparing the Eq. 5.15 and Eq. 5.16 it can be seen that the product of the DC-gain and the transfer poles are equal to the product of the Butterworth poles, therefore if the Butterworth poles are equal to the frequency of the transfer function the relationship between the two equations becomes

$$f^n = [1 - A(0)\beta] \prod_{i=1}^{n} p_i \tag{5.17}$$

Consequently the term $[1 - A(0)\beta] \prod_{i=1}^{n} p_i$ on Eq. 5.15 is the measurement of the bandwidth of the system. This term is called *loop-gain-poles product*, LP-product for short. The maximum bandwidth that a circuit could reach is given by

$$f_{n_{max}} = \sqrt[n]{[1 - A(0)\beta] \prod_{i=1}^{n} p_i} \tag{5.18}$$

Note that since poles are located at the negative part of the real axis and along the imaginary axis, the product of the poles might be negative or if the DC-gain is positive, the risk of complex roots looms. In order to avoid complex roots, the absolute value of the right side term in Eq. 5.18 is taken to yield

$$f_{n_{max}} = \sqrt[n]{|[1 - A(0)\beta] \prod_{i=1}^{n} p_i|} \tag{5.19}$$

5.3.1 Dominant Poles

The poles generated by the source, load and active devices are assumed to be overall important. Nevertheless, this assumption is not entirely true. There might be the case that one of the found poles could not have a direct influence on the bandwidth behaviour of the circuit.

For instance, in a three-pole system the lowest pole is placed several orders away from the other two affecting the attainable bandwidth of the system. Let's assume that this low pole is discarded, the system is transformed into a two-pole system and, as a consequence, the attainable bandwidth is reduced.

From the previous example it can be seen that a method to determine which poles are dominant is necessary. By using the characteristic polynomial is possible to establish a method to know the dominant poles. Recall that the Butterworth poles are named b_x and the system poles are p_x, therefore dominant poles are those for which the next expression holds

$$\sum_{i=1}^{n} p_i \geqslant \sum_{i=b1}^{bn} b_i \tag{5.20}$$

5.3.2 Maximally Flat Response

As established above an ideal frequency response of the amplifier must reach the desired bandwidth with the desired gain, as sketched in Figure 5.4.

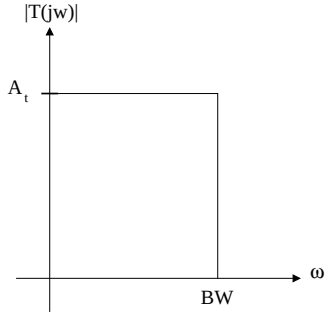

Figure 5.4: Ideal response of the amplifier.

However, when we speak about the non-ideal behaviour, maximally flat response is expected from the amplifier, as shown in Figure 5.5.

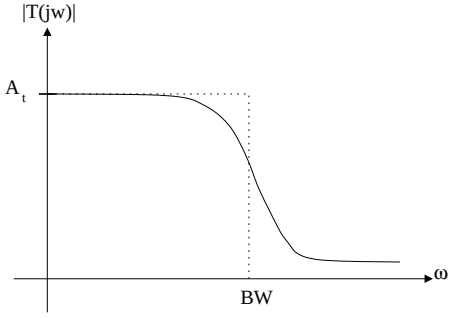

Figure 5.5: Maximally flat response.

Such response can be obtained by using a Butterworth approximation [64], with the poles located over the unit circle (for a normalised transfer function) and distributed equidistantly with symmetry on the Imaginary axis.

For a second-order system, the left-side poles are placed at

$$p_{1,2} = -0.7071 \pm j0.7071$$

for a third-order system

$$p_1 = -1$$
$$p_{2,3} = -0.5 \pm j0.866$$

For both cases all product of poles fullfils a unit bandwidth (remember, that these locations are for a normalised transfer function) and their sum equals twice the value of the real part.

5.3.3 Increasing the LP-product

When the LP-product is too low, it will not be possible to reach the specified bandwidth. It is possible to increase the LP-product by means of:

- Increasing the LP-product without adding an stage.
- Increasing the LP-product by adding an stage.

5.3.3.1 Increasing LP-product Without Adding an Stage

As it has been seen the LP-product is provided by the product of the DC loop gain and the poles in the loop. Thus, when the DC loop gain is increased without altering the product of the poles, or vice versa, the LP-product increases. It is not possible to alter in any way how the passive devices performs in the circuit. On the other hand, for the active devices (transistors) it is possible, in order to enlarge the LP-product, to modify their frequency behaviour. The f_T is the frequency capability for a transistor and can be determined by

- Type of transistor.
- Bias current.

A scheme for an increase of the f_T is shown in Figure 5.6

This process involves an exchange of the noise or clipping transistors for a better suitable device, that is, an active device with better f_T performance. The worst case scenario is the one where both devices are unable to operate within the desired bandwidth. In this case it is possible to replace both transistors in order to increase the LP-product value, the higher this LP-product value is the better performance can be achieved. This kind of compensation will not have any influence on the order of the system since this is only an exchange of devices.

Another alternative to increase the LP-product value is depicted in Figure 5.7.

Here Ic_1 refers to the collector current that fulfills the required noise behaviour for the nullor synthesis, it does not mean that it should stay in that level. In order to increase the f_T value at this stage the bias current should be increased until the maximum allowable bias

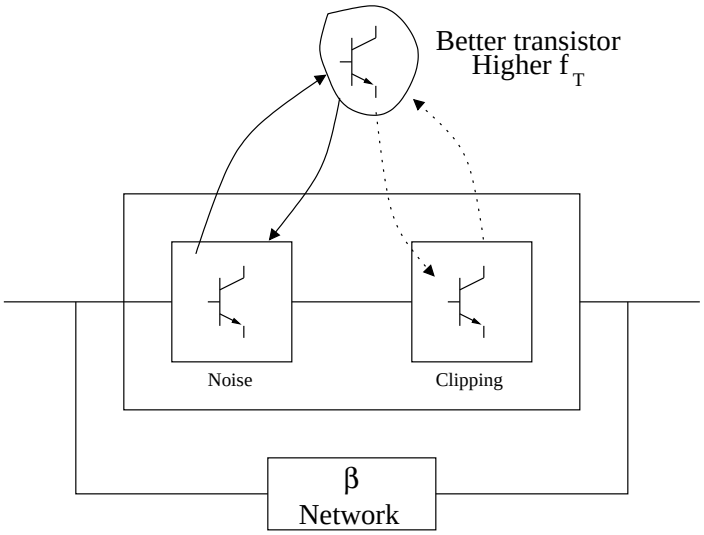

Figure 5.6: Increase the LP-product by increase of the f_T.

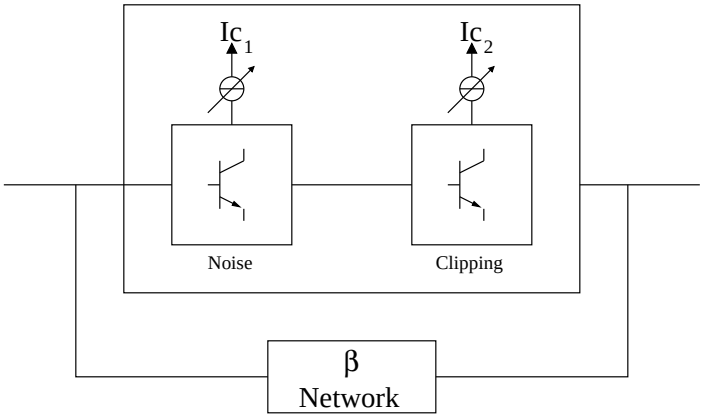

Figure 5.7: LP-product increase by an increase of the Ic.

current is achieved. Allowable bias current range is already calculated at the noise stage. In case that even at the highest bias current level the LP-product is not high enough, the next step is to modify the bias current of the clipping stage.

Bias level for the clipping stage (Ic_2) is calculated taking into account the output impedance of the amplifier and the desired output level. In order to obtain an increase on the LP-product value the active device for this stage should be increased. The maximum value to increase the bias current is limited by the maximum operating current of the device. Operating current values are found on the devices' data-sheet and is referred as *collector current — continuous* within the maximum ratings description. If after compensations within the noise and clipping stages the LP-product value is still low, addition of another stage is mandatory.

5.3.3.2 Increasing the LP-product by Adding an Stage

As already known, the order of the nullor synthesis after the design of the noise and clipping stages is two. For instance, assume that these two poles are placed at

$$\begin{aligned} p_1 &= 1 \\ p_2 &= 10 \end{aligned}$$

the DC-gain is 10, then the maximum attainable bandwidth becomes

$$f_{n_{(2)}} = LP_{(2)} = \sqrt[2]{(10)(p_1)(p_2)} = 10$$

Now let's place another device between the noise and clipping stages, and assume that this device creates a dominant pole at p_3. Therefore the LP-product calculation is given by

$$f_{n_{(3)}} = LP_{(3)} = \sqrt[3]{(DC-gain)(p_1)(p_2)(p_3)}$$

This could be rewritten as

$$\begin{aligned} LP_{(3)} &= (LP_{(2)})^{\frac{2}{3}} \sqrt[3]{p_3} \\ &= (\sqrt{p_1 p_2}\sqrt{p_3})^{\frac{2}{3}} \\ &= \left(LP_{(2)}\sqrt{p_3}\right)^{\frac{2}{3}} \\ &\approx \frac{1}{2}(\sqrt{p_3})^{\frac{2}{3}} \approx \frac{1}{2}\sqrt[3]{p_3} \end{aligned}$$

The conclusion is that if the f_T for the introduced device is higher than the bandwidth given by $LP_{(2)}$, the overall bandwidth for the system will increase.

5.3.4 The LP-product Calculation Process

Figure 5.8 shows a rather general process for the LP-product calculation, it can be explained as follows:

- The small-signal models replace the active devices within the synthesised nullor (noise, clipping and other extra stage if it was implemented). The nullator constraints are imposed on the input port of the active circuit. Therefore, the input current or voltage of the active part is zero and all the signal current flows through the feedback network.
- The loop-gain calculation is performed by breaking the loop and calculating the gain between the two open ends.
- The loop can be considered "broken" when any of the controlled sources g_{mx} is assumed to be uncontrolled. Therefore one *open end* will be located where this controlled source is placed and the other one is the controlling branch.
- Next step is to calculate the transfer function for a transconductance amplifier. This is because the input port is an independent current source and voltage is measured at the output port.
- Loop-gain is found by multiplying the transconductance transfer function by the transconductance of the open stage (g_{mx}).
- DC-loop gain can be calculated using the transfer function of the transconductance amplifier, the value is found when $s = 0$ is substituted in this transfer function.
- The poles of the system are found by taking the denominator of the transfer function and solving it for a zero value.

5.4 Frequency Compensation

Once the LP-product guarantees that the maximal attainable frequency fulfill the bandwidth needs the next step to accomplish is to place the loop poles in the Butterworth position as already explained. This poles movement can be performed by means of some compensation techniques. These techniques must not, in any way, modify the noise and clipping distortion stages. Nevertheless, as a side effect, the compensation process could decrease the LP-product value and this reduction should be keep as small as possible. It is employed the small-signal model for the active devices in order to keep calculations simple.

5.4.1 Frequency Compensation Using Root-locus

The main goal for the frequency compensation is to alter the characteristical polynomial of a system in a way that the poles of the transfer function are placed at the required positions in the s-plane. As already known, the characteristic polynomial for an $n - order$ system is given by

$$CP(s) = s^n - \sum_{i=1}^{n} p_i s^{n-1} + \cdots + LP \tag{5.21}$$

[1] provides two different frequency compensation methods. The best one that suits the automation requirements is

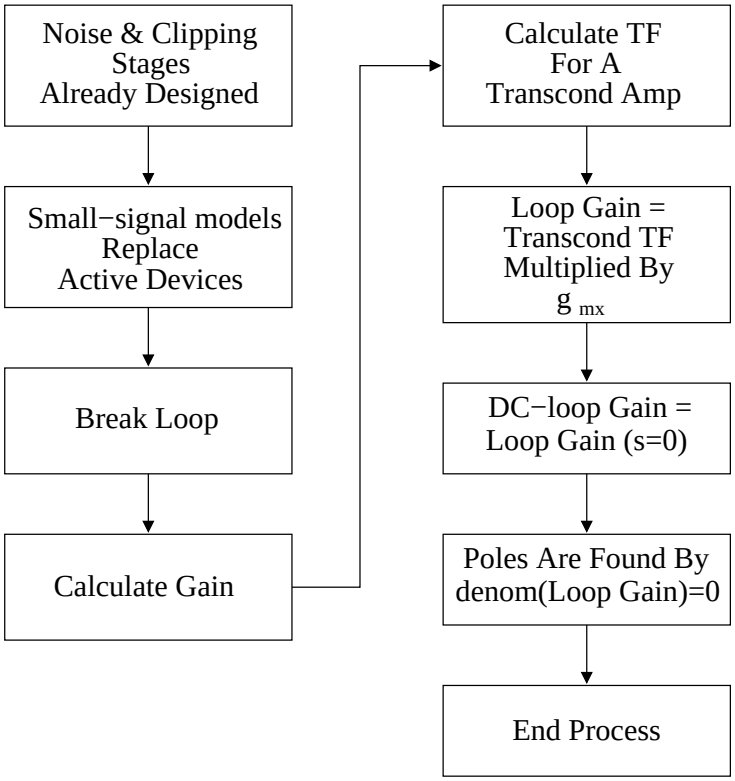

Figure 5.8: Scheme for the LP-product calculation.

- Applying the relationship between the loop-poles and system-poles. It means that compensation is performed on the loop poles. The system poles are found from the loop-poles via the root-locus method.

There are two techniques capable to shift the system poles and place them in a specified position:

1. Moving the loop poles, influencing the starting point of the root-locus.
2. Adding a zero to the loop, influencing the shape of the root-locus.

The best compensation is the one that has an influence on the shape of root-locus because by means of adding inducers or capacitors is possible to affect more than one pole, allowing to place them as close as possible to the Butterworth poles characteristic positions.

Phantom zeros compensation is the method capable to modify the shape of the root-locus and also has the property that it is only visible within the loop, this method will be thoroughly explained in the next session. Single-pole compensations are important, also, but for systems with more than two poles, their implementations become cumbersome. These will also be mentioned on the following sections.

5.4.1.1 Phantom Zeros

As already explained this technique has only an indirect influence on the total transfer via an additional pole in the system transfer. Suppose that a zero has to be created at the β

part of the loop gain $A\beta$, this zero es placed at n_1 therefore

$$\beta(s) = \beta(0)(1 - \frac{s}{n_1}) \tag{5.22}$$

then the total transfer is given by (ignoring the direct transfer A_{t0})

$$A_t(s) = A_{t\infty}(s)\frac{-A(s)\beta(0)(1 - \frac{s}{n_1})}{1 - A(s)\beta(0)(1 - \frac{s}{n_1})} \tag{5.23}$$

recalling that for the ideal transfer

$$A_{t\infty}(s) = A_{t0}(s) - \frac{\nu(s)\xi(s)}{\beta(s)} \approx \frac{1}{\beta(s)} = \frac{1}{\beta(0)(1 - \frac{s}{n_1})} \tag{5.24}$$

finally A_t is written as

$$A_t(s) = \frac{1}{\beta(0)}\frac{-A(s)\beta(s)}{1 - A(s)\beta(0)(1 - \frac{s}{n_1})} \tag{5.25}$$

As it can be seen the zero created in β is only visible in the loop $\beta(s)$ and creates an additional pole in the system transfer, however this pole can be disregarded. Therefore, when a zero is implemented in the β part of the loop it is referred as *phantom zero*. Phantom zeros can be implemented at:

- The input of the β network.
- The output of the β network.
- The input of the β network.

In order to obtain the best influence for the phantom zeros technique, the pole created by this compensation should have a higher frequency value than the phantom zero. To know how far the additional pole is from the phantom zero, this function can be calculated

$$\frac{p}{n} = \delta \tag{5.26}$$

Thus the new pole created is a factor δ apart from the phantom zero. An small value in δ means that the pole is somehow close to the phantom zero and the compensation is not effective. Given that phantom zeros are commonly placed near the band edge, the noise and clipping stages performances will be affected beyond this frequency. This is the reason that places this technique on the top of the others. [34] provides very useful hints for the three options to implement phantom zeros.

Phantom Zeros at Input

Current source.
This is the case where a shunt feedback provides a low-input impedance.

- When source impedance (Z_s) is a capacitance C_s, a series resistor R_{se} introduces a pole at
$$p = -\frac{1}{R_{se}C_s} \tag{5.27}$$
in the asymptotic gain $A_{t\infty}$ which will be an effective phantom zero if the impedance source Z_s is small with respect to the input source of the first stage (noise stage) Z_i. Two complex zeros can be created by using an additional inductance at Z_{se}.

- For the case that Z_s is a resistance, a series inductance L_{se} introduces a pole at
$$p = -\frac{R_s}{L_{se}} \tag{5.28}$$
in $A_{t\infty}$. A condition similar for the above case should be applied for an effective phantom zero.

- If Z_s is inductive, it is not possible to create a phantom zero in $A_{t\infty}$ with the aid of a series impedance.

Voltage source.
For the case that input series feedback provides a high-input impedance.

- When capacitive sources are employed, it is not possible to realize a pole in $A_{t\infty}$.

- For a resistive source, a shunt capacitance at the input port can be used for compensation. Granted that the source impedance is frequently *loosely coupled* to the feedback loop, the resulting phantom zero will not be very effective.

- An inductive source requires an input shunt resistance at the input port to create a pole in $A_{t\infty}$. Two poles can be created by placing a shunt capacitance at the input port. Again, as stated on the previous item, phantom zero(s) might be not very effective.

Phantom Zeros at Output

Voltage Output.
This is provided by a low-output impedance and realized with the aid of output shunt feedback.

- If the load impedance is a capacitance, a pole can be obtained in the asymptotic gain with the aid of an output series resistor R_{se}. The effectiveness of the pole depends on the magnitude of the impedances Z_o (impedance of the output stage), Z_f (impedance of the feedback network) and Z_L (impedance of the load).

- A resistive load impedance requires the use of a series inductance for the realization of a pole in $A_{t\infty}$. The pole is effective if accomplishes the same conditions stated on the previous case.

- The load impedance is an inductor; it is not possible that a phantom zero can be implemented with a series impedance.

Current Output.
High-output impedance is realized with the aid of output series feedback.

- Capacitive load does not allow the implementation of a phantom zero with a shunt impedance.
- Resistive load needs a shunt capacitance.
- Inductive load requires a shunt resistance.

Phantom Zeros at Feedback Network

[1] and [34] agree that the best way to implement compensation is through the network that fix the transfer parameter of the amplifier, i.e., the feedback network. In contrast to the previous cases, compensation process is related to the type of amplifier (voltage, current, transconductance, transimpedance).

Voltage Amplifier. Here the impedances Z_1 and Z_2 are assumed to be of the same type (resistors, inductors or capacitors). The asymptotic value of the voltage gain is given by

$$A_{u\infty} = \frac{Z_1 + Z_2}{Z_1} \tag{5.29}$$

- When Z_1 and Z_2 are resistors, poles in $A_{u\infty}$ can be inserted by placing a capacitance in parallel with R_1 or an inductance in series with R_2. In order to guarantee the effectiveness of the phantom zeros, it depends on the ratio between R_1, R_2 and the compensation devices (C_{ph},L_{ph}).
- For the case that R_1 and R_2 are capacitive, a pole in $A_{u\infty}$ can be obtained by inserting a resistor in series with Z_2.
- Inductive feedback network is an unpractical case, therefore no further discussion is offered.

Transadmittance Amplifier. Only one element is going to be influenced for the realization of a pole in

$$Y_\infty = -\frac{1}{Z} \tag{5.30}$$

- If Y is a capacitance, a resistor is placed in series.
- When Y is a resistance, an inductance is used in series.
- As for Y when an inductor , no compensation can be applied.

Transimpedance Amplifier Similar to the previous case where there is only feedback element, the pole to be introduced are given by a pole in

$$Z_\infty = -Z_f \tag{5.31}$$

- For the case that Z_f is a capacitor, no compensation is suitable.
- Provided that Z_f is a resistor, a shunt capacitance will do.
- When Z_f is an inductance, a shunt resistor is the adequate solution.

Current Amplifier. The asymptotic value of the current gain of the amplifier is given by

$$A_{i\infty} = \frac{Z_1 + Z_2}{Z_1} \tag{5.32}$$

it can be seen that is the same gain as the voltage amplifier.

- When Z_1 and Z_2 are resistors, poles in $A_{i\infty}$ can be inserted by placing a capacitance in parallel with R_2 or an inductance in series with R_1. In order to guarantee the effectiveness of the phantom zeros, it depends on the ratio between R_1, R_2 and the compensation devices (C_{ph},L_{ph}).
- For the case that R_1 and R_2 are capacitive, a pole in $A_{u\infty}$ can be obtained by inserting a resistor in series with Z_1.
- Inductive feedback network is an unpractical case, therefore no further discussion is offered.

5.4.1.2 Pole-splitting

This is a technique that introduces an interaction between two poles in a way that they split, Figure 5.9. Due to this splitting the sum of the poles decrease (remember that poles are negative). It is performed by placing a capacitor C_{split} between the base and collector branches on any of the small-signal model of the implemented stages as depicted in Figure 5.10.

C_{split} creates a local loop around the transistor which affects the input pole lowering its frequency by some factor. As a result, the LP-product remains almost equal while the pole p_1 is shifted downwards (lower frequency) the same factor as p_2 is shifted upwards (higher frequency). Finally, by means of the Miller approximation is possible to calculate the factor (K) that these poles split. The value for the p_1 poles is found by

$$p_{in} = -\frac{1}{r_{\pi 1} C_{\pi 1}[1 + \frac{C_{split}}{C_{\pi 1}}(1 + g_{m1} r_{\pi 2})]} = -\frac{1}{r_{\pi 1} C_{\pi 1} K} \tag{5.33}$$

From 5.33 the value for C_{split} can be found. The most effective place for the pole splitting compensation is between nodes with high voltage gain.

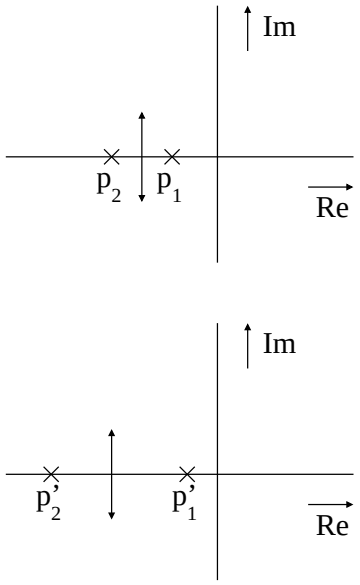

Figure 5.9: The way that pole-splitting works.

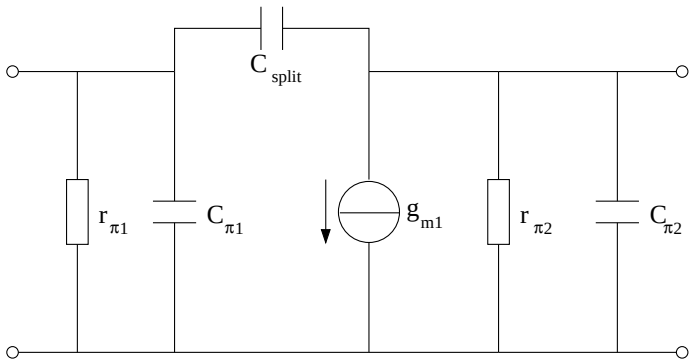

Figure 5.10: Example for pole-splitting compensation.

5.4.1.3 Pole-zero Cancellation

Pole-zero cancellation process works splitting two poles just like the pole-splitting technique, Figure 5.11. The main difference compared to the pole-splitting technique is that the compensation is performed within a selected stage, i.e., the involved stage is kept unilateral. Figure 5.12 shows the typical implementation of the pole-zero cancellation method.

As it can be seen from Figure 5.12, the influence of the C_{pz} is to shift downwards the p_1 by certain amount of frequency given by

$$p_2' = -\frac{1}{2\pi r_{\pi 1}(C_{\pi 1} + C_{pz})} \tag{5.34}$$

as for the R_{pz}, this device creates a zero placed at

$$n = -\frac{1}{2\pi R_{pz} C_{pz}} \tag{5.35}$$

When this zero is placed at the same frequency as p_2, this pole is canceled. For frequencies higher than p_2 this resistor becomes the dominant part of the compensation network and, in consequence, a pole is created at

$$p_3 = -\frac{1}{R_{pz} C_{\pi 1}} \tag{5.36}$$

Here the difference is that pole-zero cancellation keeps the involved stages unilateral. The way it works is like this: when a compensation network is added to the design a pole is shifted to a lower frequency, at higher frequencies the influence of this network is removed again and a zero is obtained. This zero can cancel other pole in the loop. Just like the phantom zero technique, an additional high-frequency pole is obtained. Effectiveness of this method is related to how much the low-frequency pole is shifted downwards and by the fact that the LP-product does not vary. Figure 5.12 shows the typical implementation of the pole-zero cancellation method.

The C_{pz} value depends on $C_{\pi 1}$ and the desired frequency to be shifted down. How much this split should be can be found assuming that the difference of p_2 and p_3 is the dominant part for the increase of the sum of the poles. R_{pz} is found just by the fact that the zero it creates has to cancel p_2. Nevertheless, On the downside it has two disadvantages: first, if a split factor of X is desired, the capacitor C_{pz} must be $X - 1$ times as large as C_{π}. Second, the effect of a lower overall loop gain is reduced.

5.4.1.4 Resistive Broad-banding

This method only acts on one pole [1], in contrast to the previous methods. On this case given that only one pole is affected and to keep the LP-product constant, the DC-loop gain is also affected. The principle works by an upward shift on a single pole. The DC-loop gain value changes in the same proportion that the pole is shifted. Therefore, LP-product value is the same.

Resistive broad-banding has two different ways to be implemented. One alternative is shown in Figure 5.13, this implementation wastes the excess overall loop gain. The original

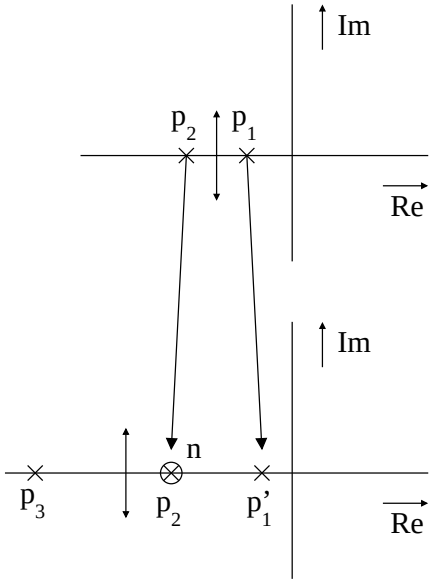

Figure 5.11: Pole-zero cancellation pattern.

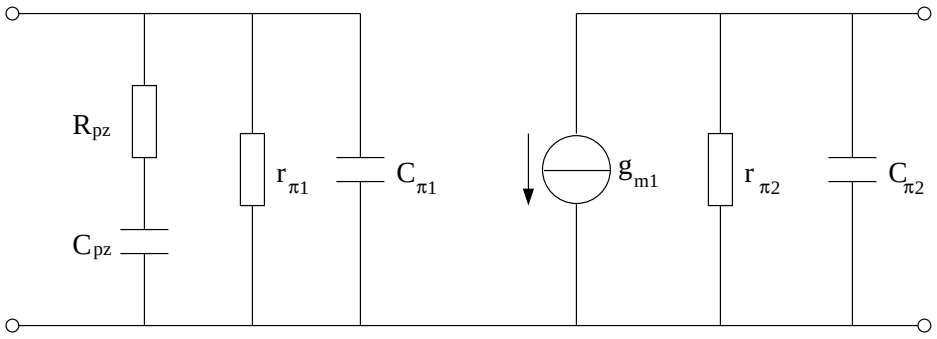

Figure 5.12: Pole-zero cancellation scheme.

pole is shifted a factor $1 + \frac{r_\pi}{R_{br}}$ upwards and the DC-gain is reduced by the same factor while the LP-product remains the same.

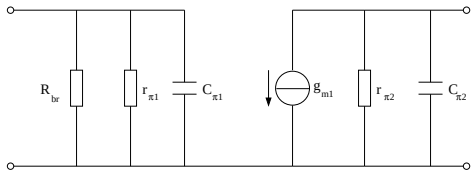

Figure 5.13: Implementation of resistive broad-banding.

Another alternative is to use the fraction of overall loop gain, the one that the overall loop gain is reduced, for linearisation purposes. This is simply a local feedback network.

5.4.2 Modify the LP-product contribution

There is a frequency compensation method that does not explicitly fit in the list of the already provided methods, is the one that changes the contribution of a single stage affecting the LP-product. Although it could be a very convenient method for certain situations.

Contribution of a single stage to the LP-product can be modified by selecting another value for the bias collector current (I_c). Since the LP-product is affected by the f_T of that particular stage, no additional frequency compensation is required or, if needed, it will be somewhat trivial. Notice that an increase on the I_c value will have an effect on the noise, clipping and distortion parameters thus a check for these parameters should be enforced.

5.5 Bandwidth Design Algorithm

The flow diagram for this stage is shown in Figure 5.14.

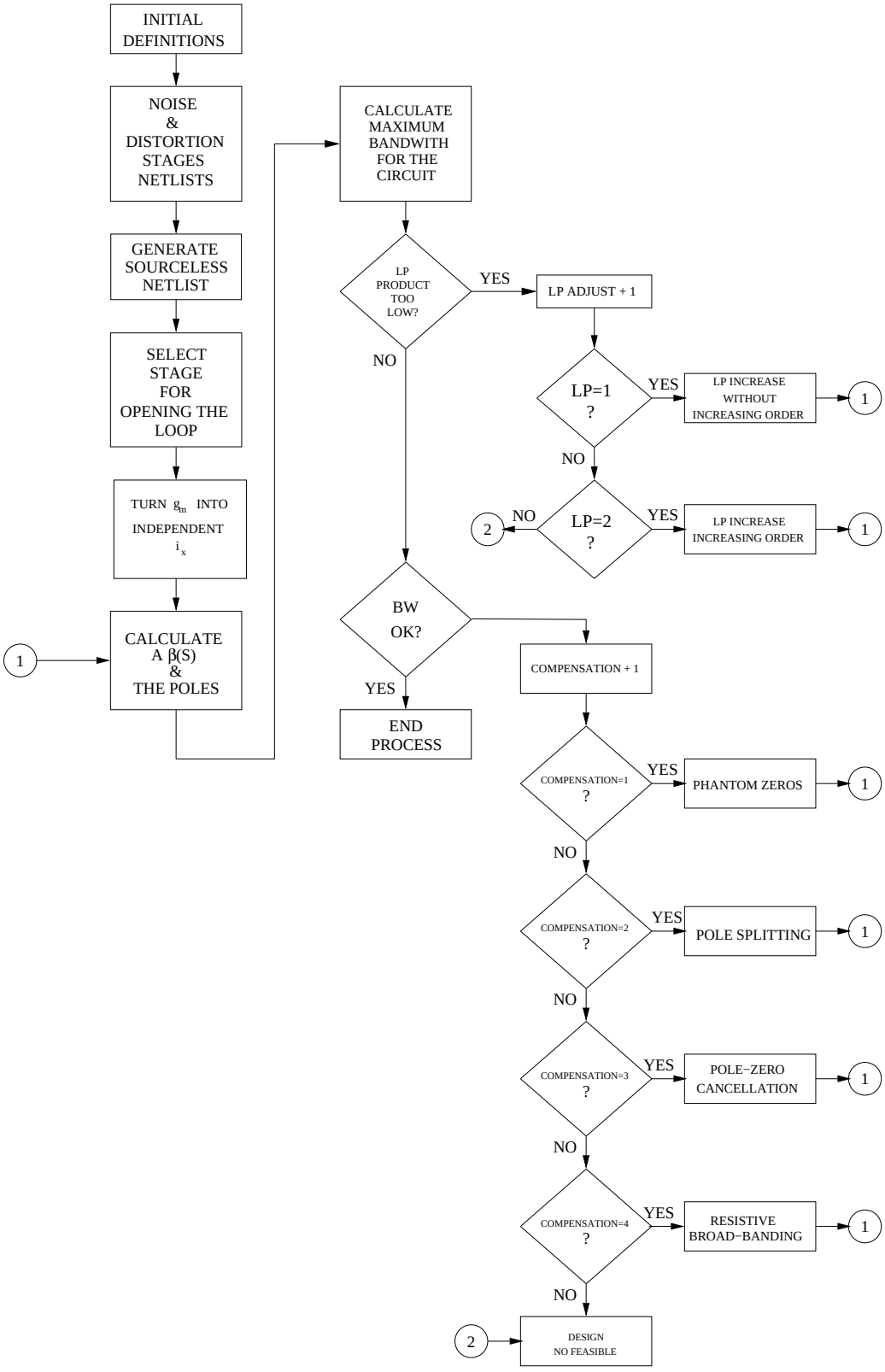

Figure 5.14: Bandwidth stage design flow diagram.

Chapter 6

DESCAD Tool Development

The design process in order to create an amplifier based on the nullor element and its subsequent synthesis has been thoroughly explained from the electronics domain point of view. In order to create an automated tool based on this methodology is necessary to change, somehow, this point of view.

Software development must be performed at this stage in order to create an useful tool that accomplishes the Structured Design guidelines without losing any basic concept. On the following sections it will be shown how the electronic design methodology is translated to the software realm. Also, the structure of this tool is presented, how calculations are performed and the approach employed to keep it simple to operate.

6.1 The Object Model

Structured Design methods evolved to guide developers who were trying to build complex systems using algorithms as their fundamental building blocks. Similarly, object-oriented design methods have evolved to help developers exploit the expressive power of object-based and object-oriented programming languages [65]. The object model has proven applicable to a wide variety of problem domains. Table 6.1 lists many of the domains for which systems exist that may properly be called object-oriented. Object-oriented design may be the only method that can be employed to attack the complexity inherent in very large systems.

6.1.1 Object-Oriented Programming

The object-oriented programming (*OOP*) is part of the object model. The main concern of the OOP is the data rather than the algorithms. It allows to organize the data within a program just like the objects in the real world. For instance, the departments of a company — sales, account, human resources, technical — or the vehicles being built in an auto factory; all of them are objects.

6.1.2 Elements of the Object Model

The basic elements where the object model rests, and as a consequence the OOP, are

Air Traffic Control	Investment Strategies
Animation	Mathematical Analysis
Avionics	Medical Electronics
Banking and Insurance Software	Musical Composition
Business Data Processing	Office Automation
Chemical Process Control	Operating Systems
Command and Control Systems	Petroleum Engineering
Computer Aided Design	Reusable Software Components
Computer Aided Education	Robotics
Computer Integrated Manufacturing	Software Development Environments
Databases	Space Station Software
Document Preparation	Spacecraft and Aircraft Simulation
Expert Systems	Telecomunications
Film and Stage Storyboarding	Telemetry Systems
Hypermedia	User Interface Design
Image Recognition	VLSI Design

Table 6.1: Object model applications.

- Abstraction - Is the capacity to inquire "something" without knowing its internal details. In an structured program, suffice it to know the specific task performed by a given procedure and is not so important *how* is performed the task [66].
- Encapsulation - Also known as *information hiding*, it is the process of hiding all of the details of an object that do not contribute to its essential characteristics [65].
- Modularity - Is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules.
- Hierarchy - It is defined as "a ranking or ordering of abstractions".

6.1.3 Relationship Between the Object Oriented Programming and the Structured Design

It is possible to stablish a relationship between the object model of the computer science and the structured design model employed in the electronics domain. The importance to relate this two models will help to decrease the complexity to translate the electronic concepts into the computer science realm.

The best approach to stablish the coincidences between these two models is to relate their basic elements and point out the similarities.

Here are the concepts, in a general way, for the object oriented programming:

- An object contains values stored in instance variables within the object.
- Thus objects contain objects to an arbitrarily deep level of nesting.
- An object also contains bodies of code that operate on the the object.

- These bodies of code are called methods.
- Objects that contain the same types of values and the same methods are grouped into classes.
- A class may be viewed as a type definition for objects.
- Analogy: the programming language concept of an abstract data type.
- The only way in which one object can access the data of another object is by invoking the method of that other object.
- This is called sending a message to the object.
- Internal parts of the object, the instance variables and method code, are not visible externally.
- Result is two levels of data abstraction.

The general concepts for the electronic structured design are:

- The design problem is divided into three smaller problems.
- These three sub-problems are assumed to be orthogonal.
- For each sub-problem further subdivisions can be performed in order to simplify as much as possible the problem.
- The nullor is an ideal object which fulfills any design constraint.
- This ideal object comprises the three problem sub-division.
- Subdivision of the problems and their implementation are not visible externally given the fact that the implementation is kwnown as "an amplifier" circuit.
- Hierarchy is employed throughout the design process.
- Simple models are defined in order to obtain a quick prediction on the feasibility of the design.

The structured design complies to the principle of the object oriented design since is a process that divide a problem in a certain number of smaller problems (in our case are three), and each of these small problems can be subdivided and so on until a specific problem is isolated.

Once the relationship to the object oriented design, the next step focuses to stablish the relationship with the object oriented programming. One good starting point is the fact that both techniques employ the *hierarchy* concept on their design guidelines, this simplifies the faceability to propose hierarchical models that could be converted in useful models. The complexity of the models should not vary, that is, simple models are employed for preliminary

tests and once these tests are succesful these simple models will be substituted with more complex models.

Structured design does not have the concept of abstaction. Nevertheless, has an element that could be employed like some kind of abstraction. As it has been explained, the nullor is where the solution of the problem starts and this element can be taken as a some sort of abstraction element. This element could be taken as **entity abstraction** [65], it means that an object represents an useful model within the problem domain. The object oriented programming focus on this matter is to employ the entity abstraction.

The modularity of the object oriented programming has its counterpart on the object oriented design in the orthogonality. The design is divided, as already explained, in three fundamental aspects and these can be also divided. Once the smaller problems have been resolved, the next step is to "assemble" once again these smaller problems until the main block is fully completed (the nullor has been synthesized). The previous explanation fulfills the definition of modularity. Figure 6.1.

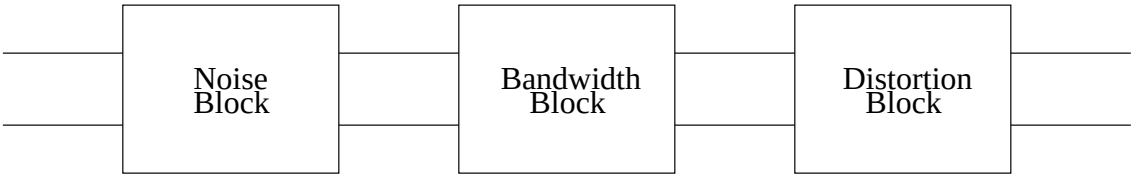

Figure 6.1: Modularity concept applied to the structured design.

The encapsulation to the structured design is not a concept that could be applied in a strict concept. Once again the nullor and its synthesis stages could fit necely to relate the concept. The nullor is an ideal element that does not exist in the real world, therefore the need to implement it using "real" devices. As the noise stage is synthesized , on the highest hierarchy point of view there is still the nullor within the amplification circuit and will be remain seen as a single block after the synthesis of the second stage. Inside of the nullor there are active devices (BJT's or MOSFET's) but in a lower hierarchy. This way it will be possible to "hide" all the object details that take place in the implementation. Figure 6.2.

6.2 Development Constraints

As it has been explained on the previous chapters there is a need of a tool capable to speed up the design process on one hand while being easy to operate on the other. This work at this point has shown how the structured design process operates and how can be related to the programming concepts of the software design. Nevertheless, there are more issues to be addressed in order to create an efficient tool. These issues are more concerned to the interaction between the software and the user. The aim of the project is to create a software not only useful to the experienced developer but to the novice student alike.

6.2.1 User Requirements

To create an useful software of any kind the software developer must take into account the needs of the user to be addressed. It is obvious that the perfect software does not

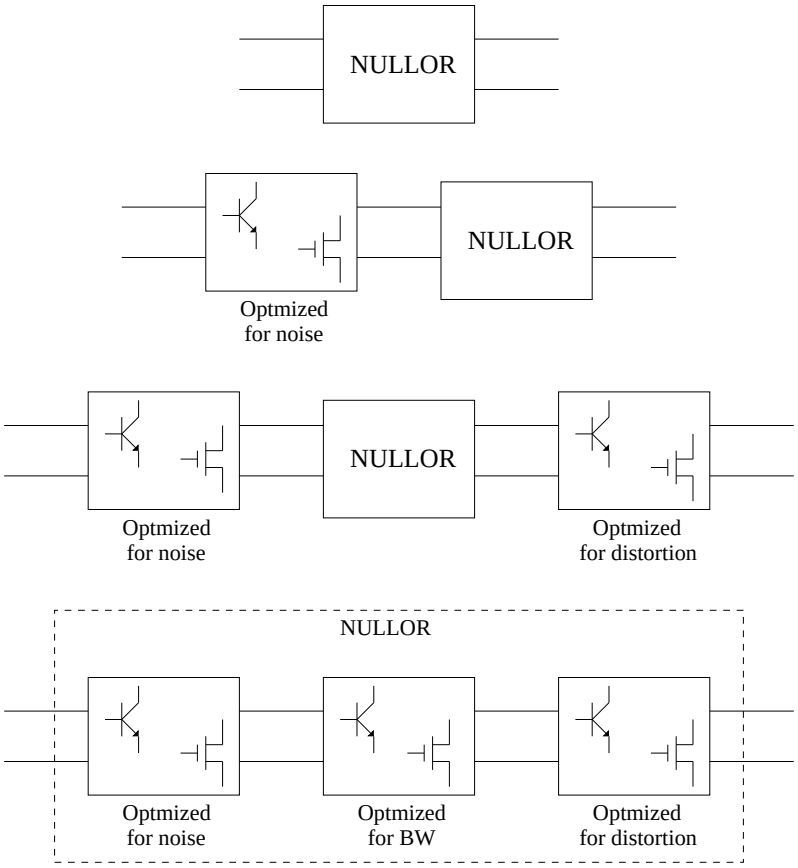

Figure 6.2: The encapsulation process in the structured design.

exist because seems impossible to fulfill the requirements of each and every potential user. Therefore, a tradeoff must be made while developing the software. The basic guidelines are already provided by the structured design methodology and the most general needs on the user side are:

- Easy to operate - This means that the operation of the software should be as simple as possible without comprising the quality of the final product.
- Beginner friendly - The tool should be implemented in a way that a novice user could be guided through the design process and achieve his goal.
- Multiple options - The simplicity lies in the freedom to have multiple choices on every stage of the design.
- Feedback - In case that an option may cause an adverse condition or the expectations may not be accomplished some kind of feedback ought to be provided to the user.
- Summary - Once the design process has finished, a detailed summary is recommended.
- Verification capabilities - One important thing for any electronic design software is the capability to provide some way to verificate the obtained results.

6.2.2 User Interface Design

The most important step towards the creation of a powerful CAD tool is the design of an adequate user interface. User interface refers to the way the information is displayed and how the user is able to interact with it. There are some basic requirements already provided that simplifies the way the user interface should be designed. There are two ways to display information on the screen (which is the default output device):

- Command Line Interface - Also known as *CLI*, this way to show the data is the simplest. Does not have any special features to interact with it but the keyboard. It is employed on situations where system resources are low or the when no special functions are needed. Given the fact that these days computer hardware is cheap and the speed of the communications are real fast, there is no need to employ this kind of interface where the aim is to reach a broad range of users. This kind of interface may not be well suited for an electronic design tool because the way to provide options to the user may be cumbersome or may cause confusion to the user. Figure 6.3.
- Graphic User Interface - Commonly referred as *GUI* is the most suitable interface to operate any kind of programs. It is the best option for new users or users with very little experience. For the electronic design concern, a GUI is an excellent option because it can provide the freedom of choice and the simplistic approach required. Figure 6.4.

There are many options to design a GUI but a detailed explanation on how it can be done is out of the scope of this work. The selected approach to design a GUI is called *wizard* [49]. The *wizard* is a GUI that guides the user by means of a series of windows, in each window there are some options to be selected and also offers the chance to return to a previous window in order to perform an adjustment.

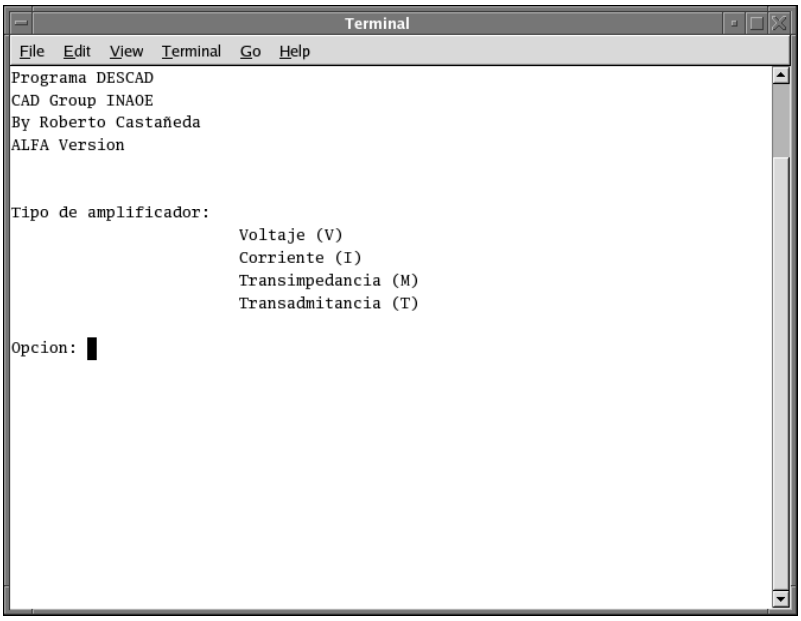

Figure 6.3: Command Line Interface (CLI) example.

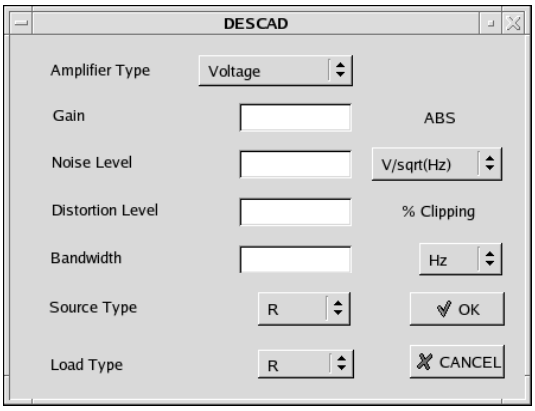

Figure 6.4: Graphic User Interface (GUI) example.

6.3 Tool Structure

In the past section it was explained that the *wizard* approach is a good option to develop the CAD tool because it follows a linear approach and is also possible to return to a previous step without comprising the development process. On the other hand, the structured design process can be seen as a linear process in a general way, Figure 6.5.

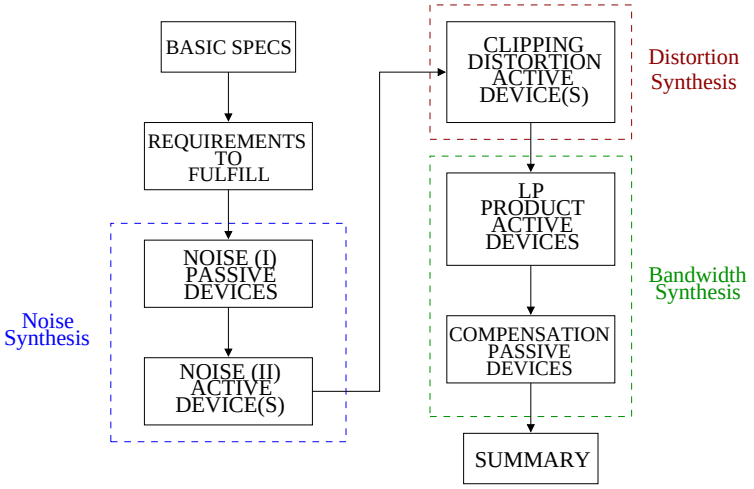

Figure 6.5: The structured design view as a linear process.

Now, it is time to consider that despite the fact that at this point of the development process the assumption that the three stages comprising the nullor synthesis have been considered orthogonal, in practical terms this assumption can no longer be effective.

6.3.1 Non-orthogonal Considerations

Given the case that once the development process has been concluded and some kind of verification process has been carried out, the results may not meet the required specifications or do not have the expected output behaviour. This result may cause some confusion because the structured design methodology **assures** that the output will be the optimum. The only way to explain this strange behaviour is given by the fact that the original assumption for the non-influence on the remaining stages could no longer be applied.

For the whole process, here are the considerations made:

- Noise Stage - At this stage of the design the noise sources come from the feedback network, source impedance and the equivalent noise sources from the active device (or devices) for the active part. It has been shown that all these quantities are already known and the noise contribution from the load can be neglected.

- Clipping Distortion Stage - It is located at the output port of the nullor. This stage depends on the output value provided by the gain and the adequate bias values (collector current and collector-emitter voltage) for the active device (or devices). At this point there are two already designed stages, however, the influence of the noise stage will not have any significant influence to the clipping behaviour of the design.

- Bandwidth Stage - The frequency behaviour of the design is based on the stimations using the noise and clipping distortion stages. For some cases another active device can be added to the design. Nevertheless, given the case that even with the addition of another stage the bandwidth is not high enough a more aggressive tactic is enforced. In order to widen the frequency range, an increase of the collector current is enforced whether it could be on the noise , clipping stages or both of them. Figure 6.6.

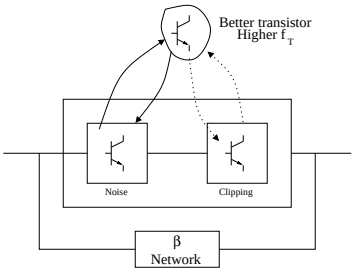

Figure 6.6: Bandwidth compensation possibilities.

Clearly, the influence of the noise and clipping stages to the overall bandwidth capability is based on the bias current for these stages. The modification, rather than refer it as an increase, of the bias value will have a direct consequence on some other constraint. An increase on the collector current at the noise stage will increase the noise generated by the device(s), the influence is in the noise and bandwidth performance of the amplifier. If the collector current is increased at the clipping stage, then the influence will be in the clipping distortion and bandwidth performance for the amplifier.

It can be seen that in order to keep one fundamental aspect within constraint (bandwidth), the consequence is that the other two (noise and clipping) will be affected in a direct way. There is a chance that even this modification could make the design non-feasible or set up another design step just to verify that the amplifier is within requirements. It can be seen that the orthogonallity can not be applied and should be taken into account in the design process and the tool development as well. Figure 6.7.

6.4 DESCAD

The developed CAD tool has been named *Descad_Wizard*, *Descad* stands for *Des*ign *CAD* and *Wizard* is to indicate that the design approach is performed by using the *wizard* approach, that is, the whole development process is performed by filling up some fields within a set of consecutive windows as already explained. Calculations are programmed in C++ [66] and the graphical user interface is provided by Qt [49].

Another tools is also employed to perform some calculations. Maple software [67, 20] is useful for complex tasks, especially for the *Chain-matrix* calculations or to obtain certain symbolic functions to later be translated to the C++ programming language.

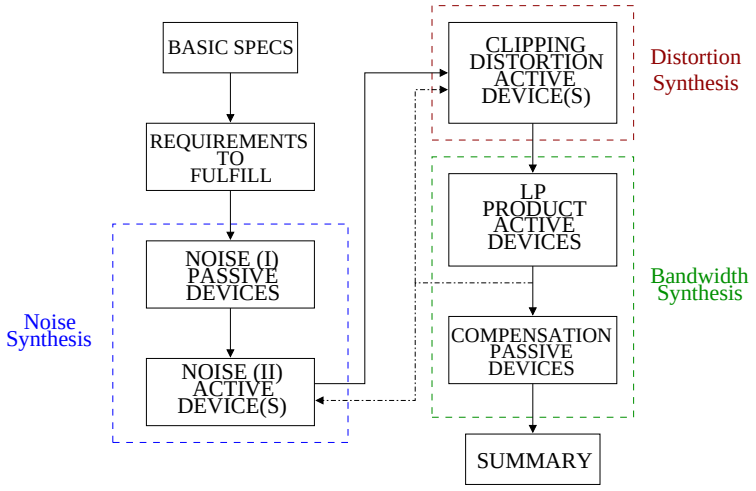

Figure 6.7: Non-orthogonal considerations on the amplifier design.

6.4.1 KMNA Maple Routine

Besides the programming performed using the C++ programming language, another code was also developed. Named KMNA, it is basically one big function written in Maple code and loaded as a library. Its purpose is to calculate the *Chain Matrix* for any given electronic circuit.

The first step is to calculate the MNA matrix from a netlist. This netlist is given in a special format known as *lists list* [20], here is an example of the netlist

$$NetList := [[P1, R1, R2], [[1, 2, 0, 4], [2, 0], [4, 2]], [N1, 1, 1]]; \tag{6.1}$$

the netlist has three parts. The first part $[P1, R1, R2]$ refers to the name of the devices, the allowed devices are (Table 6.2):

As it can be seen from the table, the devices that the library can handle are named after the SPICE [11] simulation software. For the active devices like the bipolar transistor, MOSFET and diode, these are translated to their basic models using passive devices and controlled sources.

The second part of the netlist $[[1, 2, 0, 4], [2, 0], [4, 2]]$, refers to the node numbers where the devices are placed. There are only three posibilities:

- Two-node Devices - These are devices like resistors, capacitors, inducers, and some more.
- Three-node Devices - Basically are the BJT and MOSFET devices. Note that for the MOSFET device the *bulk* terminal for these devices are assumed to be grounded, therefore not taken into account.
- Four-node Devices - The Nullor and controlled sources have four nodes.

Finally, the third part $[N1, 1, 1]$ is to put the value related to the device. As for the Nullor, nullator and norator devices there is no need to specify a numerical value; just place

Device Name	Device Identifier
Resistor	R
Capacitor	C
Inductor	L
Nullor	P
VCVS	E
CCCS	F
VCCS	G
CCVS	H
Coupled Inductors	K
Ideal Transformer	T
Nullator	N
Norator	O
Short Circuit	S
Independent Current Source	I
Bipolar Transistor (BJT)	Q
MOSFET	M
Diode	D
Independent Voltage Source	V
Impedance	Z
Admittance	Y

Table 6.2: Allowed devices for the KMNA library.

a symbolical value because it will be ignored by the program. The active devices like the BJT's and MOSFET's are described by the values of their internal Pi-models.

Once the netlist has been defined, there is still one more value to be completed. In order to process succesfully the netlist, the ports of the system are defined as follows:

$$\begin{aligned} Ports := [[&IN\_PORT\_PLUS, IN\_PORT\_MINUS], \\ &[OUT\_PORT\_PLUS, OUT\_PORT\_MINUS]]; \end{aligned} \tag{6.2}$$

The first pair of nodes embaced in square brackets are the input port nodes. The terms $PLUS$ and $MINUS$ are for reference purposes only. The second pair of nodes are to describe where is the output port. The netlist and ports definitions are stored in a file to be read by an application written in Maple. This application is capable, among other things, to calculate the *Chain-matrix* of the circuit and the *Impedance-matrix*. The main reason to create this library is the capability of the Maple software to perform complex calculations. Another useful features are that the numerical results provided by Maple can be saved in a file and further retrieved by C++ where it can be run in the background of any program.

6.4.2 DESCAD Structure

The general software structure has been already provided in the previous sections, nevertheless it is necessary to explain in detail how the process is been carried out. Therefore, for the sake of clarity, the general block diagram fot the software is shown in Figure 6.8.

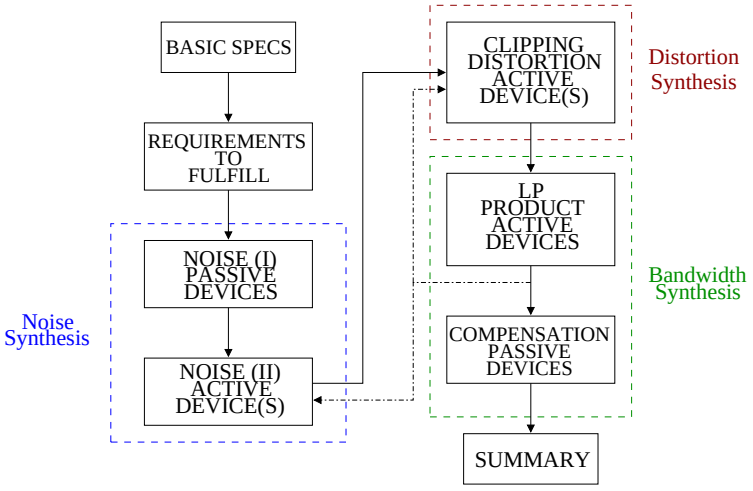

Figure 6.8: The general structure for the CAD tool.

In the following sections the DESCAD software desgin tool is going to be described and detailed in its operation.

6.4.2.1 Screen 1 - Introduction

To keep an easy operation of the software, an introduction screen is very useful. Figure 6.9.

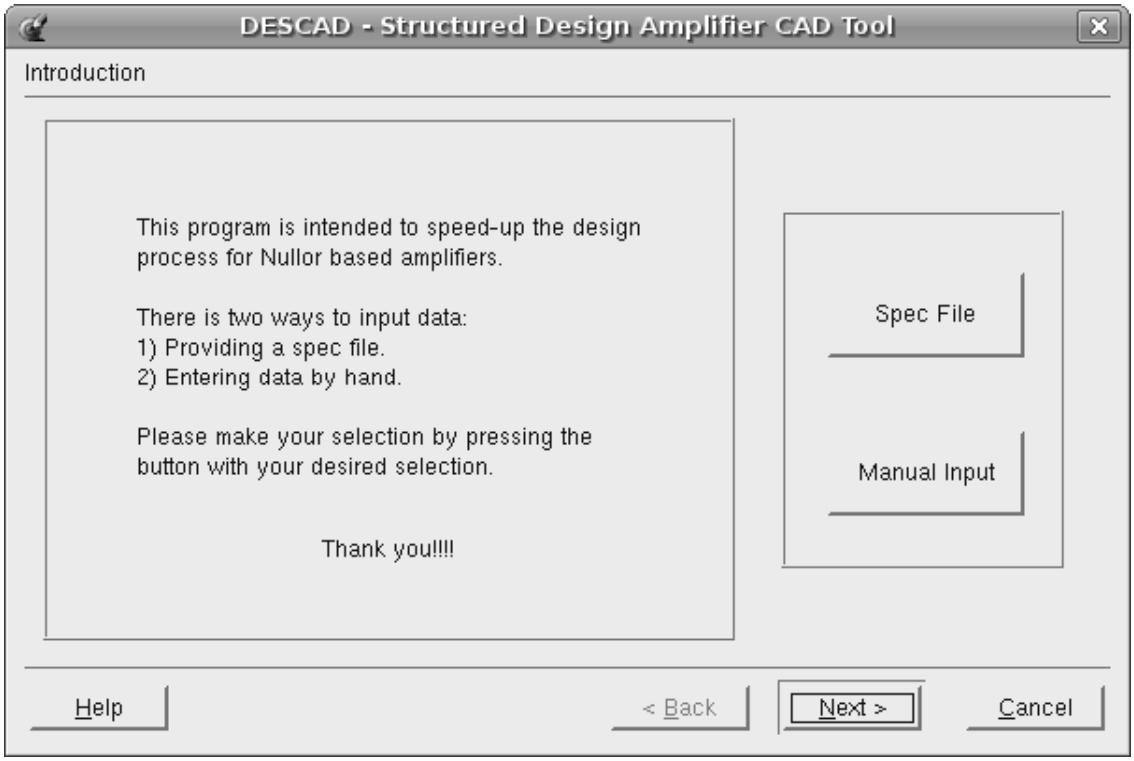

Figure 6.9: DESCAD wizard screen 1, the introduction.

It is a very simple screen where offers to the user two options to introduce the data. The first option button provides a file where the basic definitions for certain amplifier design are stored, these definitions are given in the following format (Table 6.3):

- Amplifier Type - The allowed values are : **0** - Voltage Amplifier, **1** - Transconductance Amplifier, **2** - Transimpedance Amplifier, **3** - Current Amplifier.
- Configuration - Values are: **0** - Single Loop, **1** - Double Loop B.
- Gain - It is possible to provide a value in scientific format, i.e. 0e0, in decimal format, i.e. 0.000, or an integer value.
- Phase - It could be: **0** - Positive output phase, **1** - Negative output phase.
- Source Impedance Type - The options are: **R** - Resistor, **C** - Capacitor, **L** - Inductor.
- Source Impedance Value - Just like the gain value, it is possible to provide the value in scientific format, decimal or integer.
- Load Impedance Type - The options are: **R** - Resistor, **C** - Capacitor, **L** - Inductor.
- Load Impedance Value - Value can be provided in scientific, decimal or integer format.
- Noise Unit - It could be: **db** - which means that the noise value is given in *decibels*, **spectral** - means that the noise value is given in the units of volts over square hertz

Definition	Value Type
Amplifier Type	Integer
Configuration	Integer
Gain	Real
Phase	Integer
Source Impedance Type	Character
Source Impedance Value	Real
Load Impedance Type	Character
Load Impedance Value	Real
Input Signal Value	Real
Noise Unit	String
Noise Value	Real
Positive Clipping	Integer
Negative Clipping	Integer
Bandwidth	Real

Table 6.3: Definition for the input file archive.

$(V/\sqrt{Hz})$ if the source is a voltage source or in amperes over square hertz $(A/\sqrt{Hz})$ for a current source.

- Noise Value - It could be provided in decimal or scientific notation.
- Positive Clipping - This value is the percentage that the *positive* part of an AC signal can be *chopped*, the limit is 100% that means that the signal can be clipping distorted completely.
- Negative Clipping - It is based on the same principle applied to the previous point, the difference here is that applies to the negative part of the AC signal.
- Bandwidth - This value is given in scientific notation.

In order to avoid any confusion, the extension to load this kind of file is *spc*. Creation of the spec file is done using any text editor and placing **just one value per line**. An example of the format for the file is given in Table 6.4:

For the second option button, it indicates that the input values are goind to be performed by filling out some required data using some windows. Basically, the user will provide the same information as the specs file but with graphical aid.

6.4.2.2 Screen 2 - Basic Data I

This window is presented in Figure 6.10 and it has been labeled with numbers for a quick reference. The figure shows the default selections in certain options.

1. Amplifier Type - There are four options to select (in order): Voltage (default), Transconductance, Transimpedance, Current.

2
0
1e6
0
R
15e3
R
1e3
.5e-6
db
10
0
0
1.5e6

Table 6.4: Basic specs definition file example.

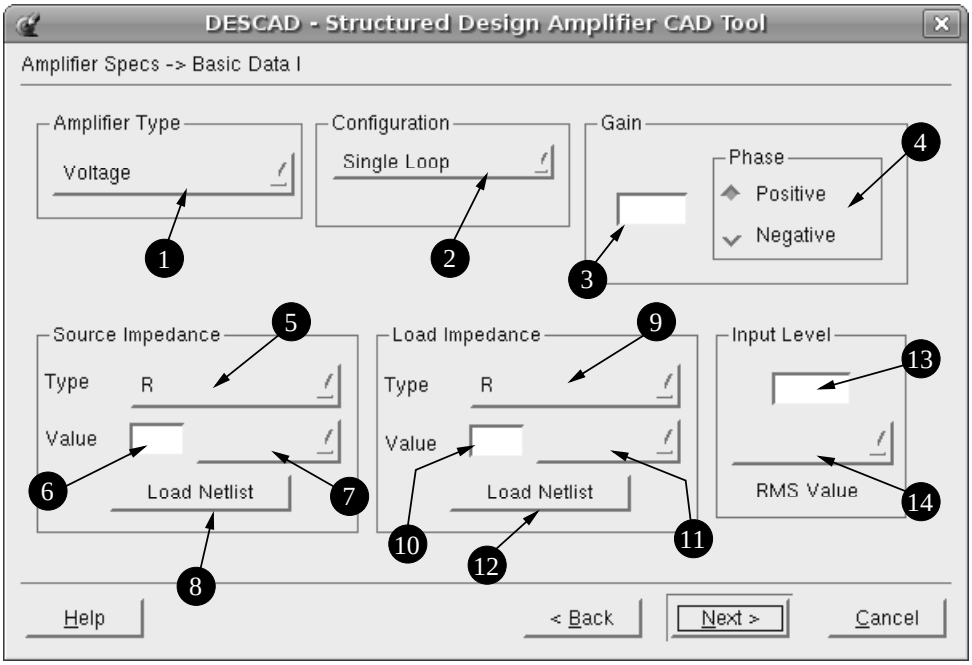

Figure 6.10: DESCAD wizard screen 2, basic data definition.

2. Configuration - It refers to the feedback network topology to be designed. The options are: Single Loop (default) or Two Loop B.

3. Gain Value - This can be supplied in decimal or scientific format.

4. Gain Phase - For the output value, it can be selected a positive or negative phase.

5. Source Impedance Type - It could be one of these options: R (Resistor), C (Capacitor), L (Inductor) or User_Defined. The User_Defined option allows to activate de Load Netlist button, this option will be explained below.

6. Source Impedance Value - The value can be provided in integer or decimal formats.

7. Source Impedance Units - This selection has a direct reference to the type of source impedance. The possible values to be selected are presented in Table 6.5, Table 6.6 and Table 6.7.

Resistor
Ohms
kOhms
MOhms

Table 6.5: Resistor units that can be selected.

Capacitor
F
mF
microF
nF
pF

Table 6.6: Available unit options for the Capacitor.

Inductor
H
mH
microH

Table 6.7: Options to select for the Inductor units.

8. Source Impedance Load Netlist - Once the User_Defined option is selected in the Source Impedance Type option, this button becomes active. It allows to load a Spice-like format file (Figure 6.11).

   An example of this file is given in Table 6.8

   Note that the main difference to a Spice netlist is that here the input and output ports are explicitely defined. For a Spice netlist there is no need to define such ports.

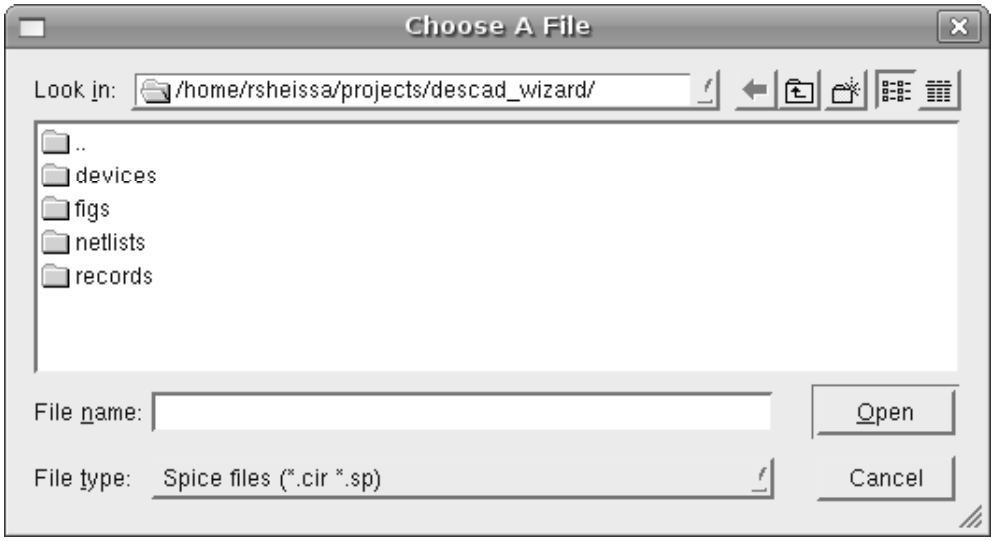

Figure 6.11: Dialog to select a file for the Load Netlist option.

```
* Netlist for notes'
* output impedance
rl 1 0 10e3
cl 1 0 100e-12
ports 1 0 1 0
.end
```

Table 6.8: Contents of an example file for the User_Defined option.

9. Load Impedance Type - The options for this selection are the same as the Source Impedance Type.

10. Load Impedance Value - It can be supplied in decimal or integer values, just like the Source Impedance Value.

11. Load Impedance Units - Just like the Source Impedance Units, the possible options are provided in Table 6.5, Table 6.6 and Table 6.7.

12. Load Impedance Load Netlist - The same principle for the Source Impedance is applied to this case, that is, this button is activated once the User_Defined option is selected. The load file window is activated (Figure 6.11). Finally, the format of the file is similar to the one shown in Table 6.8. Do not forget to include the definitions for the input and output ports when creating the file.

13. Input Level Value - It can accept an integer or decimal value.

14. Input Level Units - This value depend on the selected Amplifer Type. For the voltage source amplifiers the options are shown in Table 6.9 and for the current source amplifiers are depicted in Table 6.10. Note that the provided value is an RMS [62] value.

Voltage Source
V
mV
microV
nanoV

Table 6.9: Voltage source input level values.

Current Source
A
mA
microA
nanoA

Table 6.10: Current source input level values.

Once all values have been properly selected, the process moves forward to the next screen.

6.4.2.3 Screen 3 - Basic Data II

The aim of this screen is to stablish the constraints that must be satisfied by the design. It is shown on the Figure 6.12.

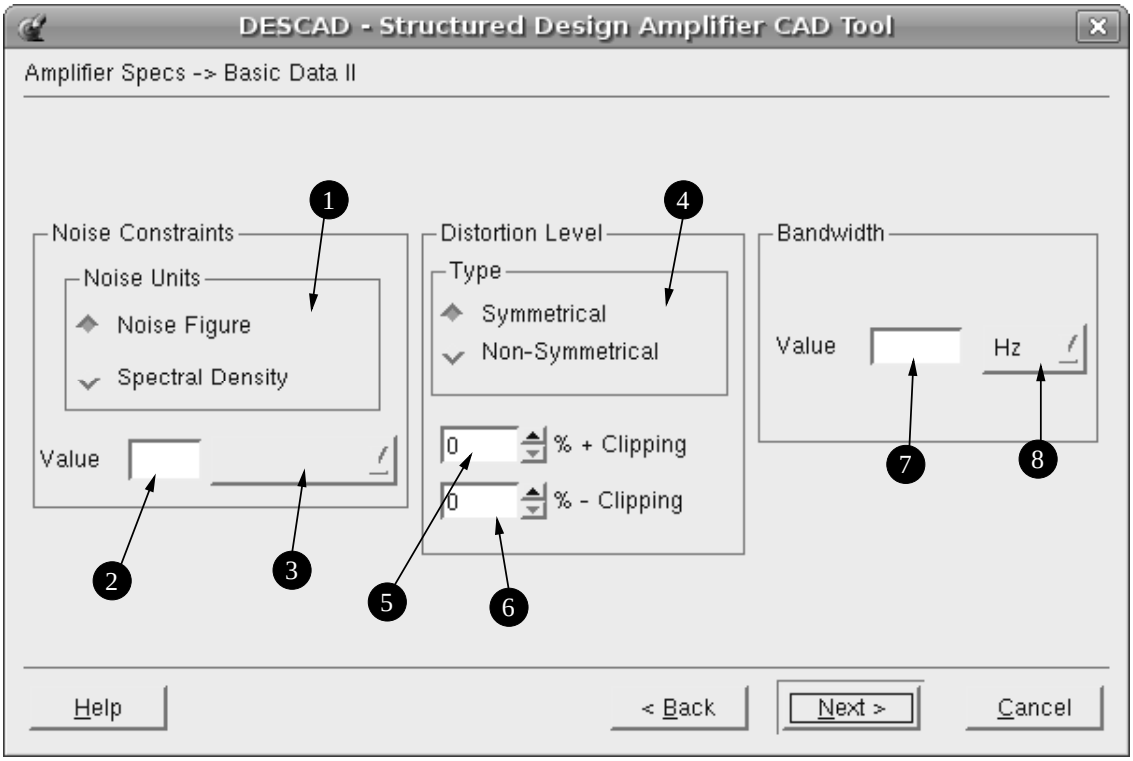

Figure 6.12: DESCAD wizard screen3, amplifier constraints definition.

1. Noise Units - It is possible to select from two units how the noise is going to be defined. The first choice, this is the default, is named *Noise Figure*; the second choice is to provide the noise specs in terms of the *Spectral Density* units. These selections remain valid while the source impedance is defined as a resistor, otherwise the noise must be provided in terms of the *Spectral Density*.

2. Noise Value - For this option, it is possible to provide an integer value or in decimal format.

3. Noise Units Definition - Given the case that the Noise Units selector is the *Noise Figure* button, there is no need to provide any factor by which the Noise Value should be multiplied because the value is in *decibels*. Nevertheless, when the selector is the *Spectral Density* button, there are two options provided in Table 6.11 and Table 6.12.

Voltage Source
V/sqrt(Hz)
mV/sqrt(Hz)
microV/sqrt(Hz)
nanoV/sqrt(Hz)
picoV/sqrt(Hz)

Table 6.11: Spectral Density definition units for a voltage source.

Current Source
A/sqrt(Hz)
mA/sqrt(Hz)
microA/sqrt(Hz)
nanoA/sqrt(Hz)
picoA/sqrt(Hz)

Table 6.12: Spectral Density definition units for a current source.

4. Distortion Level Type - Distortion can be defined in a symmetrical way or in individual components. The button can select any of the two possibilies labeled as *Symmetrical* and *Non-Symmetrical*.

5. Positive Clipping - When the *Non-Symmetrical* option is selected, the allowed clipping of the positive part of the signal is determined by this control. The 0% of clipping means that no clipping is desired for the positive part of the output signal. On the opposite case, a 100% of clipping is the highest value of clipping, in other words, allows clipping even if the signal is completely "chopped".

6. Negative Clipping - Like the previous point, this refers to the amount of clipping distortion that the designer allows for the negative part of the output signal. Again, the 0% means no clipping allowed while the 100% is the highest distortion allowed.

7. Bandwidth Value - The third and last consideration to design the amplifier is specified here. This value can be an integer or a decimal number. It is important to keep in mind that the value entered must be equalled or exceeded by the designed circuit.

8. Bandwidth Multiplier - This pull down menu allows to select the frequency multiplier for the *Bandwidth Value* defined in the previous point. In the Table 6.13 are displayed all the available options.

Frequency	Multiplier
Hz	×1
kHz	×1000
MHz	×1000000
GHz	×1000000000

Table 6.13: Frequency multipliers reference table.

6.4.2.4 Screen 4 - Noise Synthesis (Step 1)

Like the title of the section suggests, the synthesis of the nullor has just begun. At this stage some calculations are performed in an automated fashion and visual feedback is supplied for reference purposes. Figure 6.13.

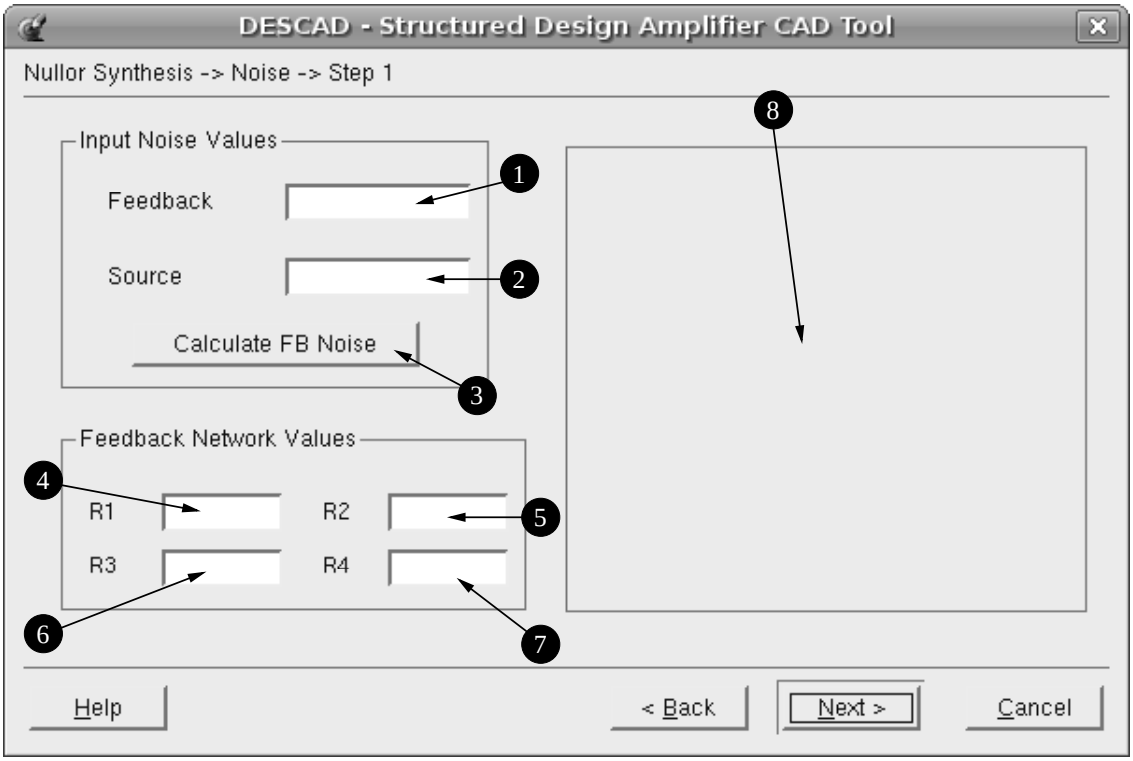

Figure 6.13: DESCAD wizard screen 4, the begin of the nullor synthesis.

1. Input Noise Values, Feedback - In this line the calculation of the noise produced by the feedback network is displayed. This is just visual feedback and the shown value cannot be modified. The value shown is given in *Spectral Density* units.

2. Input Noise Values, Source - The noise generated by the source impedance is shown here. Only resistors can generate noise due its characteristics [34], therefore, in the case of capacitive or inductive sources the value to be displayed is zero. Like in the previous point, this information is only visual feedback and it cannot be modified. This value is given in *Spectral Density* units.

3. Calculate FB Noise - Once this button is pressed, it initiates the calculation of the noise generated by the source and feedback elements. The sum of the noise components for the source and feedback network elements is then substracted to the noise constraint provided in the previous screen; the resultant number becomes the noise to be accomplished by the nullor synthesis.

4. Feedback Network Values R1 - Here the value for the resistor that accomplishes both gain and noise requirements is displayed. For the feedback networks that contain just one element, the value of the gain is displayed in this box.

5. Feedback Network Values R2 - This box is only employed in the single loop voltage and curren amplifiers, and in the double-loop amplfiers. The value of the resistor that provides the adequate gain and noise is showed.

6. Feedback Network Values R3 - Not active.

7. Feedback Network Values R4 - Not active.

8. Nullor Topology - This empty canvas is employed to show the topology of the selected amplifier configuration. It is provided as a visual aid to the designer.

6.4.2.5 Screen 5 - Noise Synthesis (Step 2)

In the previous screen the noise to be accomplished by the active part has been defined. The nullor device is going to be implemented by means of an active device. Given the characteristics of this kind of devices it is not possible to place one in the design without the generation of certain amount of noise. In Chapter 3 the formulas to calculate the approximate noise generated by this kind of devices were presented. In fact, these formulas are the base for the noise synthesis. This wizard window is shown in Figure 6.14.

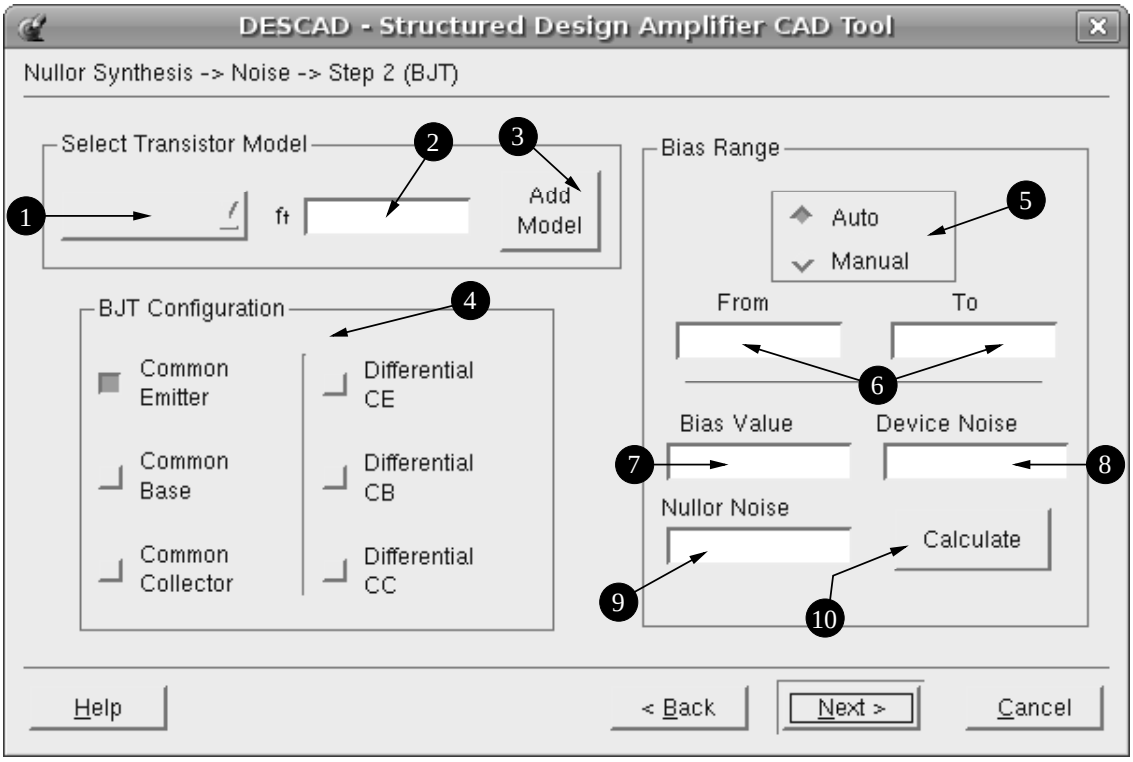

Figure 6.14: DESCAD wizard 5, active device implementation.

1. Transistor Model List - This is a pull-down list that shows several transistor models to be chosen from.

2. Transistor Ft - Here it is displayed the maximum bandwidth that the active device is capable to handle provided by the data sheet of each device. It is provided just as a guide because the maximum frequency of the device relies on various factors [60].

3. Add Model - In case that certain model does not appear in the list it can be possible to add it. By pressing the button another form pops-up to fill it with the required parameters. Figure 6.15.

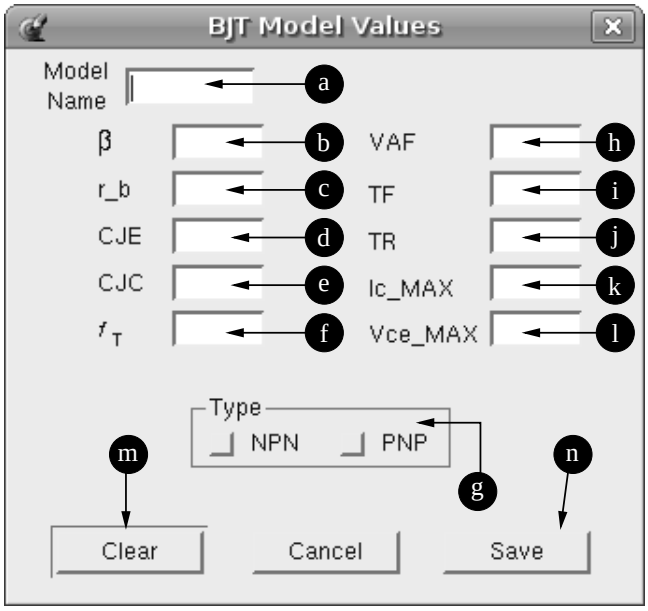

Figure 6.15: BJT new model definition form.

(a) Model Name - This is self explained, the user can provide any name for the model to be defined.

(b) β - The value of the beta that is provided in the spec sheets for the transistor or in Spice model netlists.

(c) r_b - Means the value of the internal base resistance. It can be found on the Spice model netlists or in some general data sheets.

(d) CJE - The zero bias Base-Emitter depletion capacitance employed to calculate the value of C_π.

(e) CJC - The zero bias Base-Collector depletion capacitance employed to calculate the value of C_μ.

(f) f_T - It is the maximum bandwidth that the device can handle according to the spec sheet.

(g) Type - It could be either *NPN* or *PNP*.

(h) VAF - Forward Early voltage. Alternate name is VA.

(i) TF - Ideal forward transit time. This value is employed in the calculation of the value of C_π.

(j) TR - Ideal reverse transit time. This value is employed in the calculation of the value of C_μ.

(k) I_C_MAX - This is value referenced as *Collector Current — Continuous* in the data sheet. It is the maximum value for the bias current, any value beyond this maximum limit may cause internal damage to the device.

(l) V_{ce}_MAX - In the data sheet information is provided as *Collector - Emitter Voltage* V_{CEO}. It means the maximum bias voltage that the device can handle, any more voltage beyond this value may cause severe damage to the device.

(m) Clear - If any mistake in the input data is detected, by pressing this button all the fields are erased and must be filled once again.

(n) Save - Once all the fields are adequately filled, this button saves the information in a file for later retrieval by the tool.

4. BJT Configuration - It is possible to select one single device or a combination called *differential*. Differential means that two transistors are place in a certain position that could benefit in a certain manner the design outpu capabilities. This kind of configuration has already been discussed in previous chapters. All the posibilities are displayed in the most simple way in order to avoid confussions.

5. Bias Range - There are two ways to calculate the adequate bias value in order to be within the noise constraint for the active device(s). The most simple is the automatic way (this is the default selection), and by a manual selection of the bias value.

6. From and To Values - When the manual calculation is selected, the range within the device can operate **below** the noise value is displayed. The lowest value is shown in the *From* field and the highest in the *To* field.

7. Bias Value - If the automatic selection is kept for the bias calculation, the optimum (and lowest) collector current value is shown. For the manual case, the user most provide a value that lies within the *From* and *To* range. If the user provides any invalid value a warning message appears indicating this situation.

8. Device Noise - The noise generated given the bias current in the *Bias Value* line is displayed here.

9. Nullor Noise - It is provided as a reference to compare the noise provided in the *Device Noise* line.

10. Calculate - This button starts the calculation of the adequate bias current when the automatic option is selected. For the manual case once an adequate bias value is provided, by pressing the *Calculate* button the noise generated by the active device is calculated and shown in the *Device Noise* line.

6.4.2.6 Screen 6 - Clipping Distortion

The noise stage is designed and synthesized. Now, the last stage will be designed and synthesized. Here the clipping conditions defined at the beginning of the process will play a significant role. Since this stage relies only on the characteristics of the active device, there are no need to calculate values for passive devices. Figure 6.16.

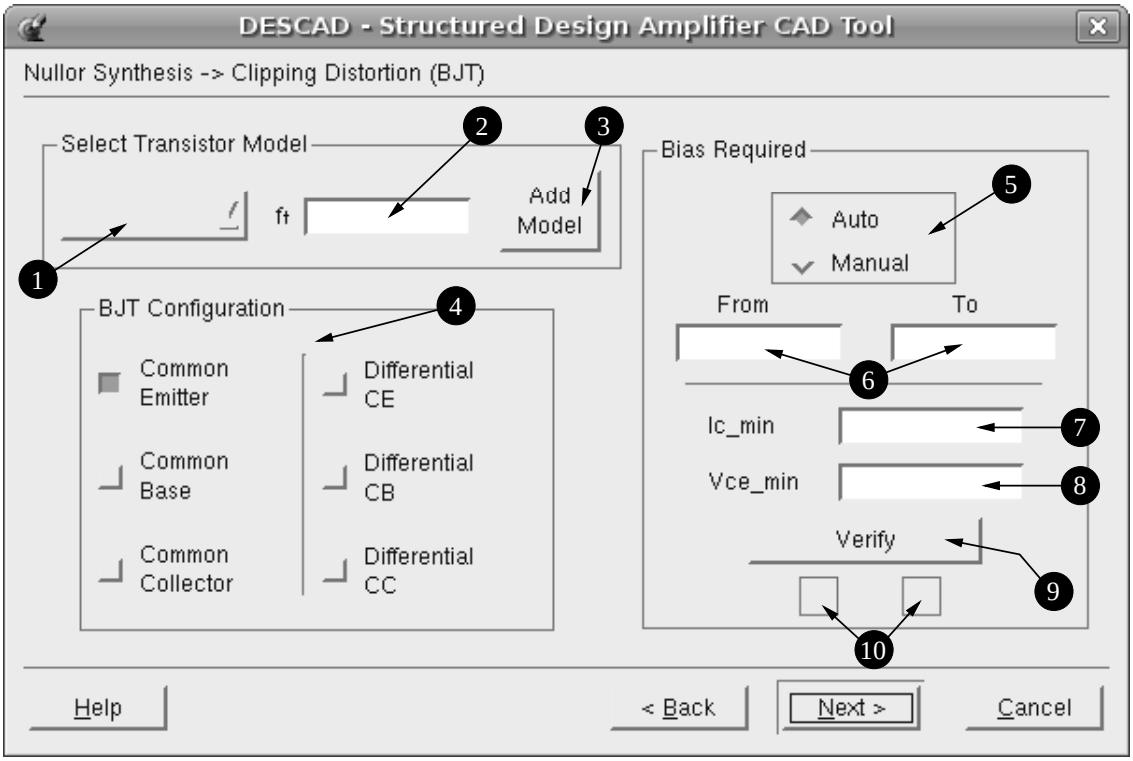

Figure 6.16: DESCAD wizard 6, clipping distorion synthesis.

1. Transistor Model List - The displayed list is the same as the one shown in the previous screen.

2. Transistor Ft - Just provided as a very general information about the maximum frequency that the device can handle.

3. Add Model - The process is the same as the one in the previous screen. Refer to the past description about the new model definition.

4. BJT Configuration - It is possible to select single devices or differential devices, the layout of the options is the same as in the noise synthesis.

5. Bias Required - The calculation for the required bias in order to avoid the provided clipping limit can be performed in two ways: the first is an automatic calculation (the default) and the second is by selecting an adequate bias value in manual mode.

6. Bias Range - Here is shown a range where the active device can accomplish **above** the desired clipping percent. The lowest value is shown in the *From* line and the highest in the *To* line.

7. I_C_min - When the automatic calculation is performed, the lowest value to avoid clipping is displayed. For the manual case, the user must supply a value for the voltage and transimpedance amplifiers. Note that the value must be within the valid range.

8. V_{ce}_min - In the case of the automatic calculation the lowest voltage bias value that avoids the clipping is shown. In the manual case the user supplies a value that lies within the valid range for the transadmittance and current amplifier cases.

9. Verify - This button starts a procedure that verifies that the found or supplied values are adequated for the selected active device.

10. Visual Output - This means that in order to show that the device is capable to operate given the voltage and current bias, a semaphore like shows in this boxes. A green light (shown in the left box) means that the device is capable and a red light (shown in the right box) means that the device is uncapable to operate under these voltage and current conditions.

6.4.2.7 Screen 7 - LP-product

At this point the orthogonality is no longer valid as it was explained previously, therefore a decision must be taken regarding the stage where some kind of compensation should be taken or if another active stage ought to be placed. The *LP-product* calculation is a very useful tool provided by the structured design methodology to obtain an estimation of the bandwidth handle capability of the design. This process, as it was thoroughly explained in the previous chapter, needs to perform certain calculations based in a very important decision that is where to break the loop. Figure 6.17.

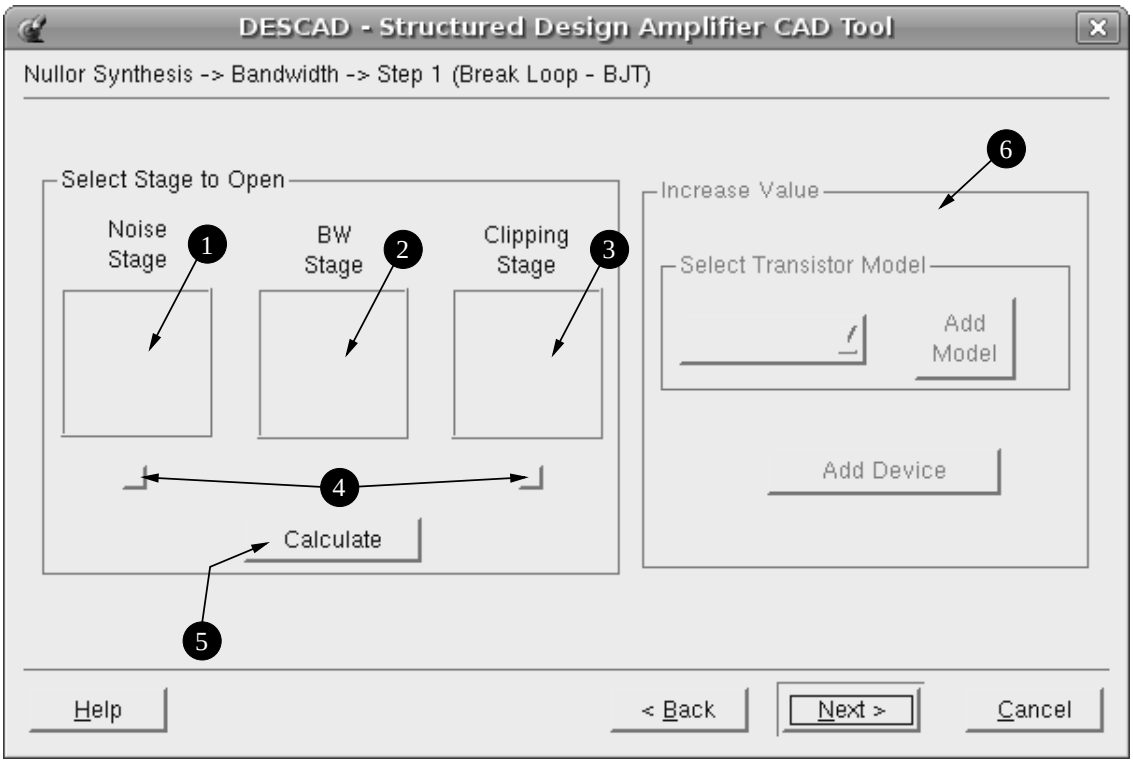

Figure 6.17: DESCAD wizard 7, breaking the loop for the LP-product calculation.

1. Noise Stage Picture - This empty canvas will depict the transistor configuration selected for this stage. This is provided for reference reason.

2. BW Stage Picture - Basically, this canvas will show a pair of wires since this stage is still under development.

3. Clipping Stage Picture - Like the Noise Stage Picture, this canvas shows the configuration selected to synthesize the stage. This image is shown for reference reason.

4. Loop Break Selection - The designer/user has the possibility to select where the loop should be broken. It is possible to select whether the noise or the clipping stages.

5. Calculate Button - By pressing the button it starts the calculation of the *LP-product*. At the end of this process the results are provided in another window (Figure 6.18).

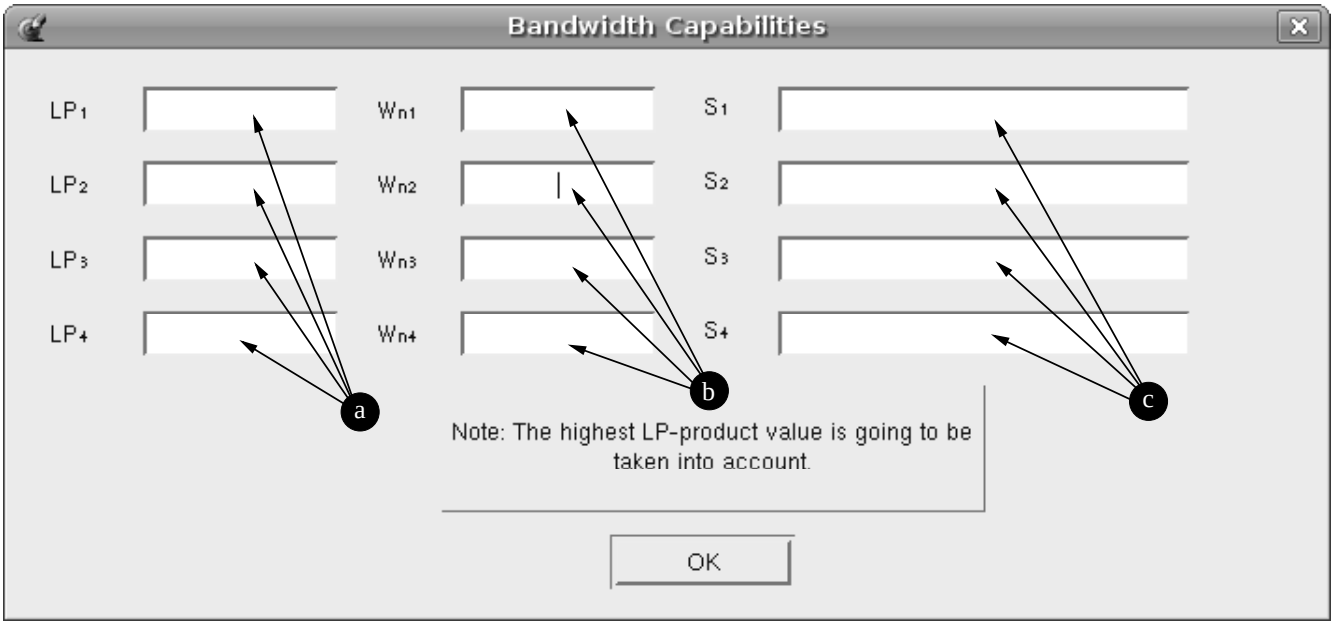

Figure 6.18: Output of the LP-product calculations.

(a) LP_x - In this column are presented all poles found by the calculation. They are sorted from the lowest to the highest valued. It is important to remember that all poles are negative.

(b) W_{nX} - This column shows the bandwidth value that the design can handle based on the LP-product value, it depends on the pole position and the number of the poles. Refer to the bandwidth chapter for further details.

(c) S_x - Every line provides the status of the pole, here is displayed if the pole is dominant or if it is characteristic can be non-dominant.

Because the highest LP-product value is taken into account, it could be possible that the predicted bandwidth does not suffice the requirement. Therefore, there is a need

for compensation in order to achieve the desired range. The noise stage is the first to be compensated by means of an increase of its bias current. If after three (3) attempts the bandwidth does not increase enough, the clipping stage is compensated. Again, three attempts are allowed to compensate this stage. Finally, if after these compensations the bandwidth is not high enough, another stage must be placed.

6. Additional Stage - This option is activated when the compensation of the two previous stages fails. It is possible to select any device from the pull-down list, just like the other stages.

6.4.2.8 Screen 8 - Butterworth Position

At this point the noise behaviour is below the desired limit, the clipping level is within the acceptable constraints and the overall bandwidth performance is high enough. The methodology at this point only requires the output frequency to have a flat response. The flat response is provided by means of placing the system poles in Butterworth position [1] as it has already explained in the previous chapter. This screen summarizes the way to perform this compensation. Figure 6.19.

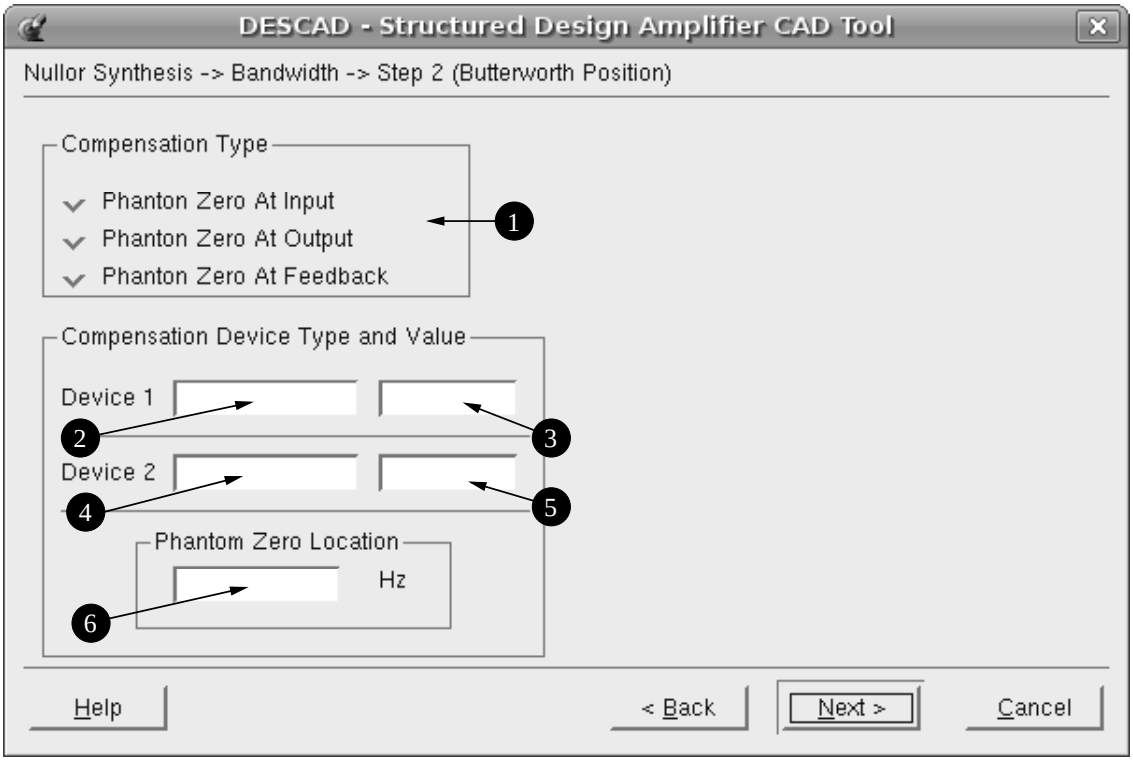

Figure 6.19: DESCAD wizard 8, Butterworth compensation.

1. Compensation Type - There are three options: 1) Phantom Zero At Input, this refers that the compensation will take place at the input port of the amplifier. This option can be applied when the input impedance is a *Capacitor* or a *Resistor*, for the *Inductor* or

an arrengement of two or more passive devices there is no way to modify the placement of the poles. 2) Phantom Zero At Output, the compensation process is executed at the output port of the amplifier. This operation can only be executed if the output impedances are a single *Resistor* or a single *Inductor*, any other variation on this topologies does not qualify to be compensated. 3) Phantom Zero At Feedback, since the feedback network is only comprised of resistors, the compensation can always be applied.

2. Compensation Device 1 Type - Here the type of the compensation component is displayed. The components could be *Resistors*, *Capacitors* or *Inductors* it depends on the configuration and source impedance.

3. Compensation Device 1 Value - The calculated value for the device to cause the movement of the poles.

4. Compensation Device 2 Type - In case that the calculation is performed in the feedback network with two devices. The type of the device depends on the configuration of the amplifier and the components as well.

5. Compensation Device 2 Value - Shows the required value for the device.

6. Phantom Zero Location - For reference purposes is displayed the position where the phantom zero is located on the negative part of the frequency axis.

6.4.2.9 Screen 9 - Final Summary

This is the final screen to be presented to the user. Here a summary of the constraints are displayed and the results for every step in the design, also. It is supplied in a way that is very easy to understand, even for the non-experienced users. As an added value a netlist in Spice-like is saved in a file as a means to verify that the design behaviour is satisfactory. Figure 6.20 shows the last window of the wizard.

In the following chapter some examples are provided to show how the tool performs under varying design conditions.

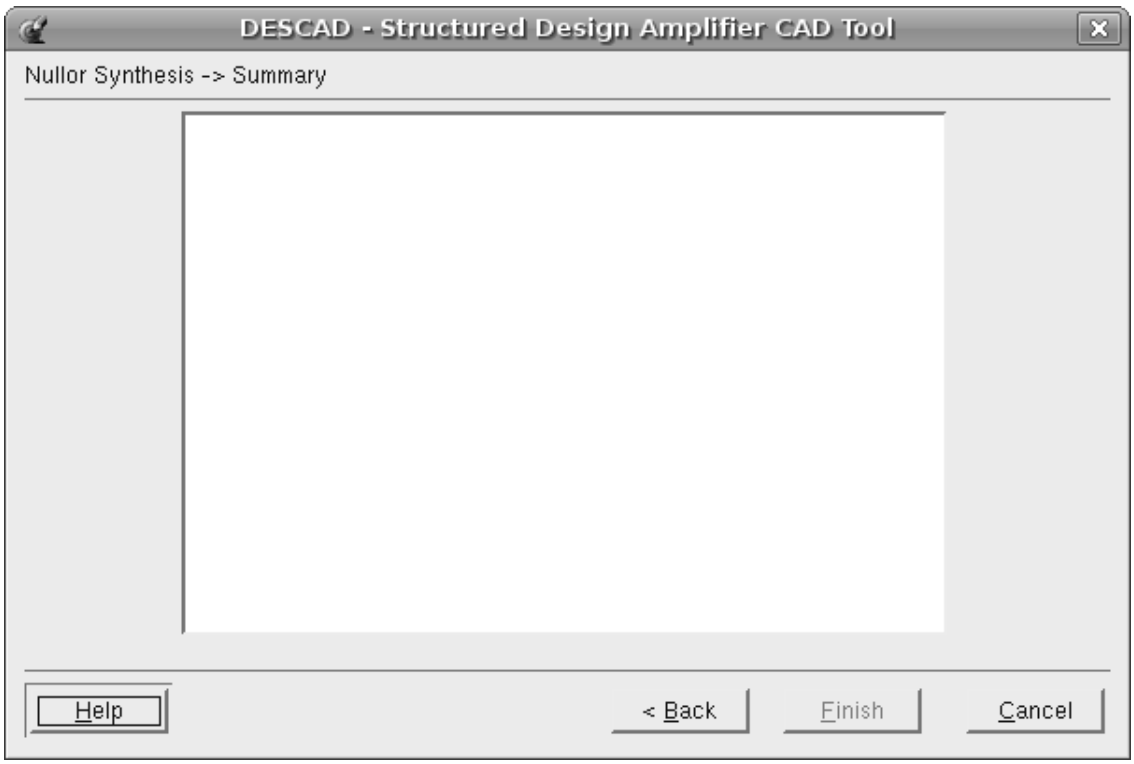

Figure 6.20: DESCAD wizard 9, final summary window.

Chapter 7

Design Examples

In this chapter some examples will be provided and their results reviewed to validate the accuracy of the developed CAD tool. As a quick reminder, the feedback network can only be resistors but the input and output impedances can be any passive device (capacitor, resistor or inductor) or a more complex arrangement.

For comparison purposes a transimpedance amplifier is the first to be presented because the result will be compared to an already design presented in [1]. The next proposal is a voltage amplifier in a single loop configuration. This configuration will serve to test the capability of the tool for the adequate calculation of the feedback network based in two devices. Finally, a single-loop current amplifier will show the capacity of the tool for a more complex design.

All the results will be verified in a circuit simulator. By performing circuit simulation is possible to obtain the noise performance of the circuit, the numerical placement of the poles and zeros and the frequency behavior of the signal.

7.1 Transimpedance Amplifier Test Case

The amplifier to be designed, at this moment including the nullor, is shown in Figure 7.1. It is a single-loop topology that is controlled by the value of the single feedback element. For the purposes of this work the feedback element is a resistor. The load and source impedances can be implemented by either a single element or an arrangement of passive elements, as it has been already explained in previous chapters.

This amplifier should comply with the following requirements (Table 7.1):

Regarding the noise requirement, in the example found in [1] provides the noise requirement in terms of *SNR (Signal to Noise Ratio* therefore a transformation was performed to translate this value into the *Spectral Density* value that the tool recognizes. The value provided in the example is 70 dB, then the transformation was performed as follows:

$$70 = 20log_{10}\left(\frac{0.5}{V_{out-noise}}\right) \tag{7.1}$$

Solving to find the output noise level, the Equation 7.1 becomes

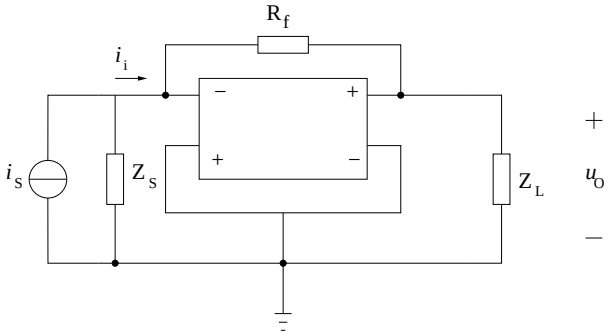

Figure 7.1: Transimpedance amplifier scheme including the nullor.

Input Level	0.5μ Amperes
Source Impedance	10 nF
Output Level	0.5 V
Output Impedance	10 kΩ ‖ 100pF
Input Noise Level	158 pA/Sqrt(Hz)
Clipping Allowed	0 %
Bandwidth	1.5 MHz

Table 7.1: Design specifications.

$$V_{out-noise} = \frac{Vout}{exp\left(\frac{70*ln(10)}{20}\right)} = 0.1581138830e-3 \tag{7.2}$$

Finally, to find the equivalent input noise value it is necessary to divide the value found in Equation 7.2 by the gain:

$$Gain = \frac{Output - value}{Input - value} = \frac{0.5}{0.5\mu} = 1e6 \tag{7.3}$$

$$V_{in-noise} = \frac{V_{out-noise}}{Gain} = \frac{0.1581138830e-3}{1e6} = 0.1581138830e-9 \approx 158e-12 \tag{7.4}$$

After the requirements have been established, the design is depicted with the changes in Figure 7.2. Now, the development tool is loaded and the manual input option is selected as shown in Figure 7.3.

Some of the requirements are typed in the second screen shown in Figure 7.4.

The constraints for the design are entered in the third screen depicted in Figure 7.5.

At this stage, the following information is displayed on the screen (Figure 7.6).

The nullor synthesis commence with the noise calculation of the passive components. Given the fact that the source impedance is a capacitor, the noise contribution is equal to zero. The Figure 7.7 shows the results of the calculations.

Once the noise generated by the passive devices is known, the total noise contribution is subtracted by the noise value found and the remaining amount will be fulfilled by the

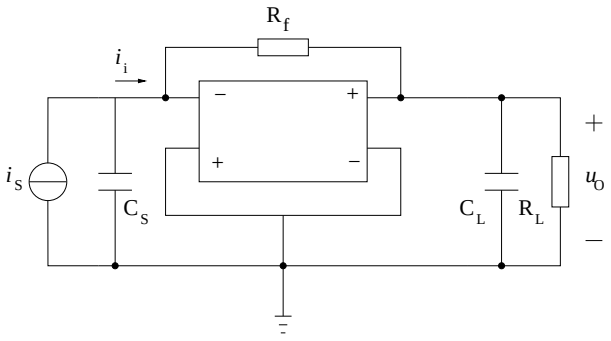

Figure 7.2: The desired configuration of the negative-feedback transimpedance amplifier.

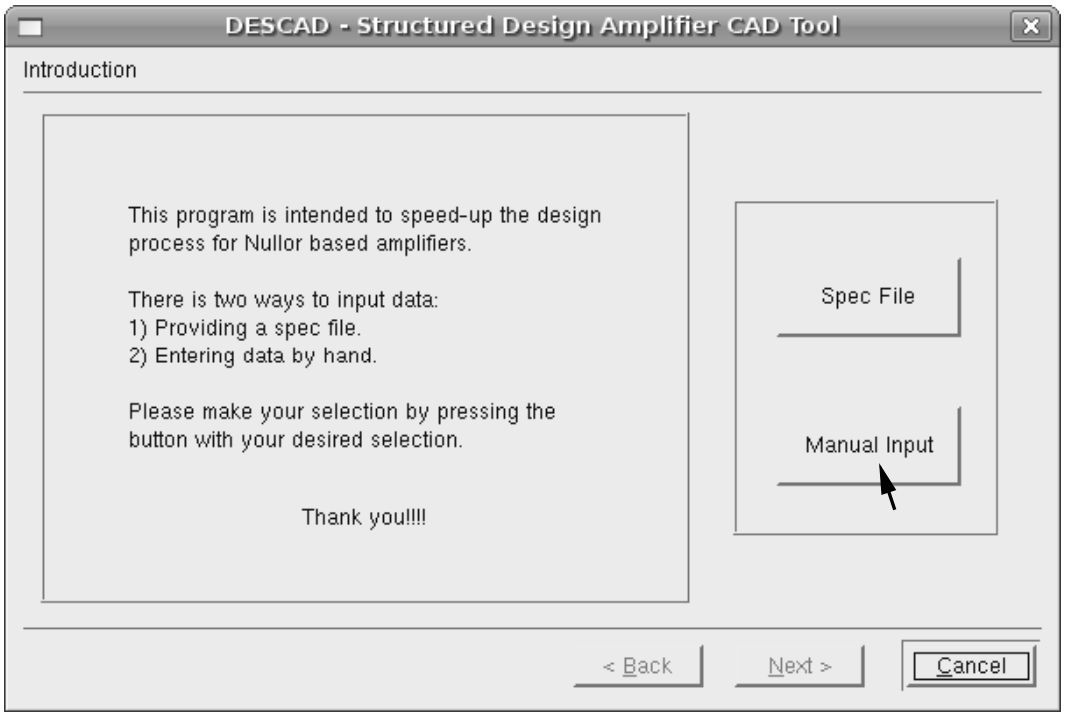

Figure 7.3: The first screen of the tool. The selected option is *Manual Input*.

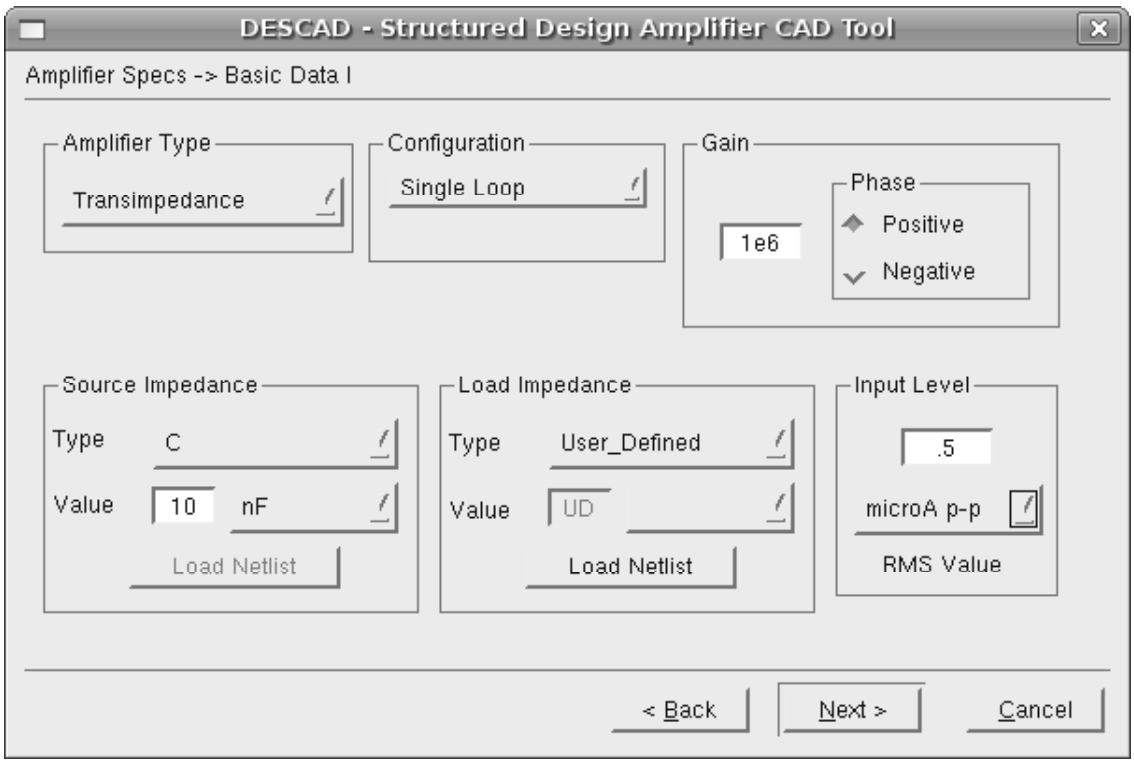

Figure 7.4: The basic requirements are typed in the second screen.

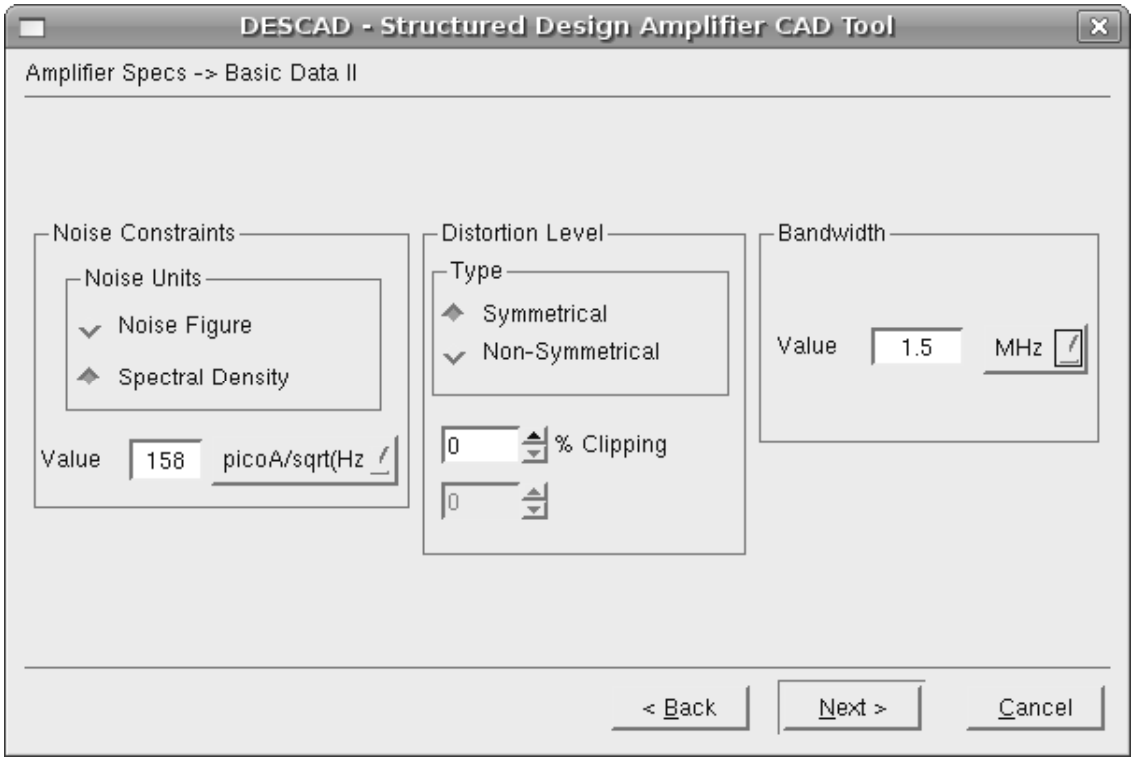

Figure 7.5: The circuit constraints are placed in this screen.

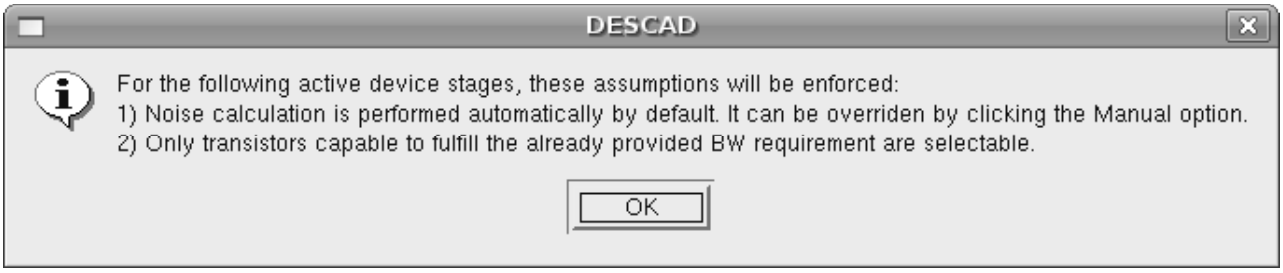

Figure 7.6: Before continuing to the next stage, some important information is displayed.

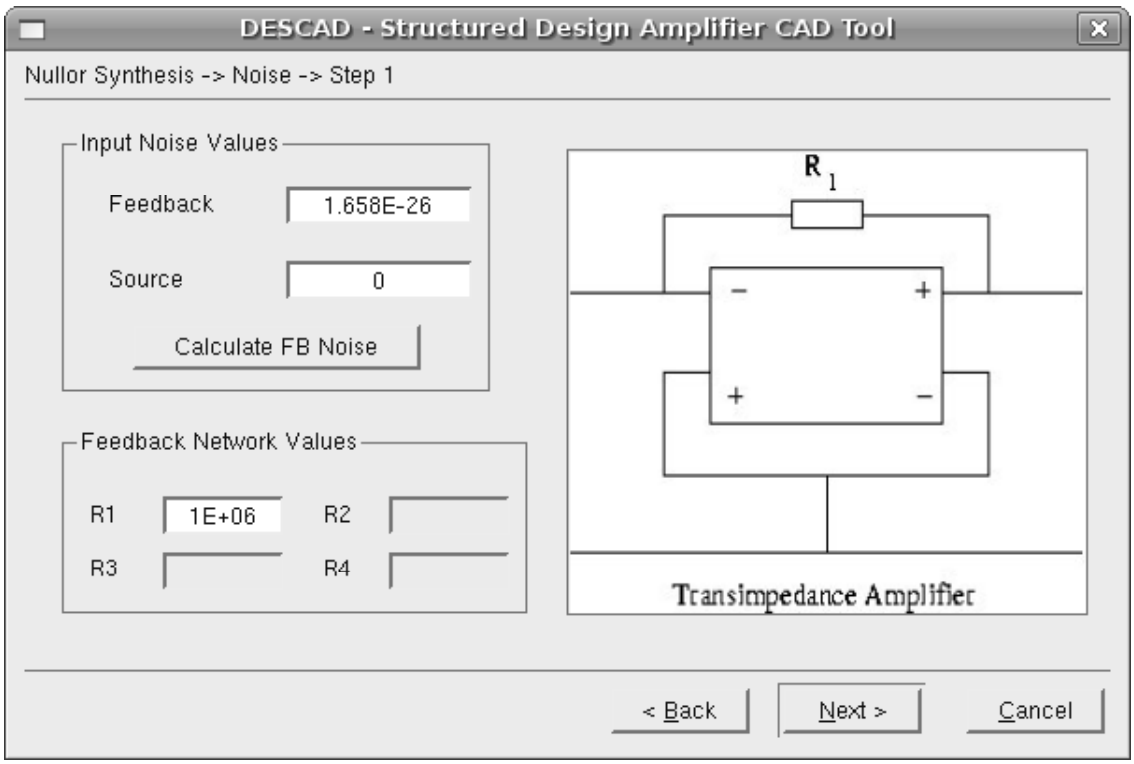

Figure 7.7: Results of the noise contributions by the passive elements in the design.

active device (or devices) employed on the noise stage of the nullor synthesis. The selected model is a 2N3904 in a differential common emitter configuration, within this windows the minimum bias current that accomplishes the noise limit and the noise generated are shown. For reference reasons the noise value to be accomplished is also shown. The results are found in Figure 7.8.

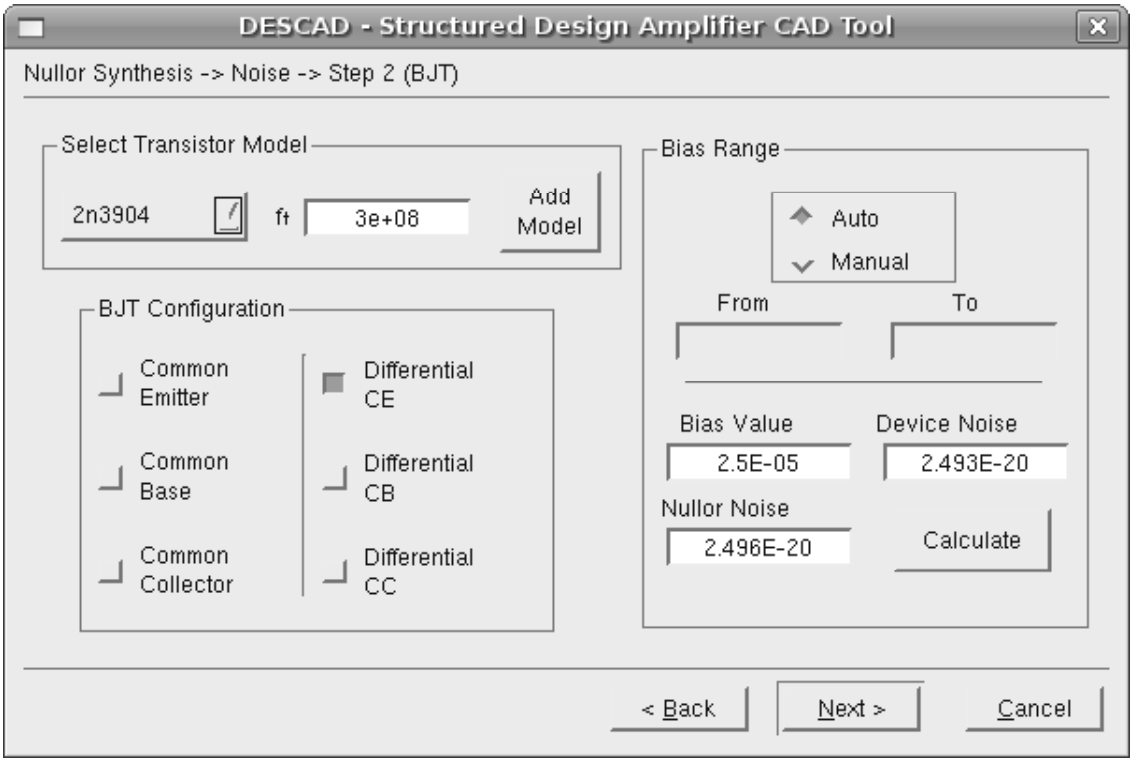

Figure 7.8: The active device to be placed at the noise stage is selected so is the desired configuration.

The next step is the clipping distortion design. According to the specs provided, no clipping is allowed because the value is zero percent. Therefore, the calculations will show the bias values for the collector current and the collector to emitter voltage in order to avoid the clipping effect at the output port of the synthesized nullor. The selected active device is the same as the previous stage, that is, the 2N3904 BJT transistor in the common emitter configuration. In Figure 7.9 the minimum values are shown. The calculated values are compared to the maximum allowed values of the BJT transistor according to the manufacturer datasheet.

At this design stage two out of three stages are already designed and the most crucial one is about to be calculated. If the desired bandwidth cannot be achieved there would be no reason to continue with the design process. Here the designer can select which stage will be "broken" and perform the LP-product calculation. The Figure 7.10 shows where the loop is broken. Once the calculations are finished the screen in Figure 7.11 comes up to show the poles of the system, where are located and the result of the LP-product for each of them. These poles are shown in progressive order, that is, the poles are ordered from the higher to the lowest located pole. Given the fact that these poles are negative, the pole closest to the

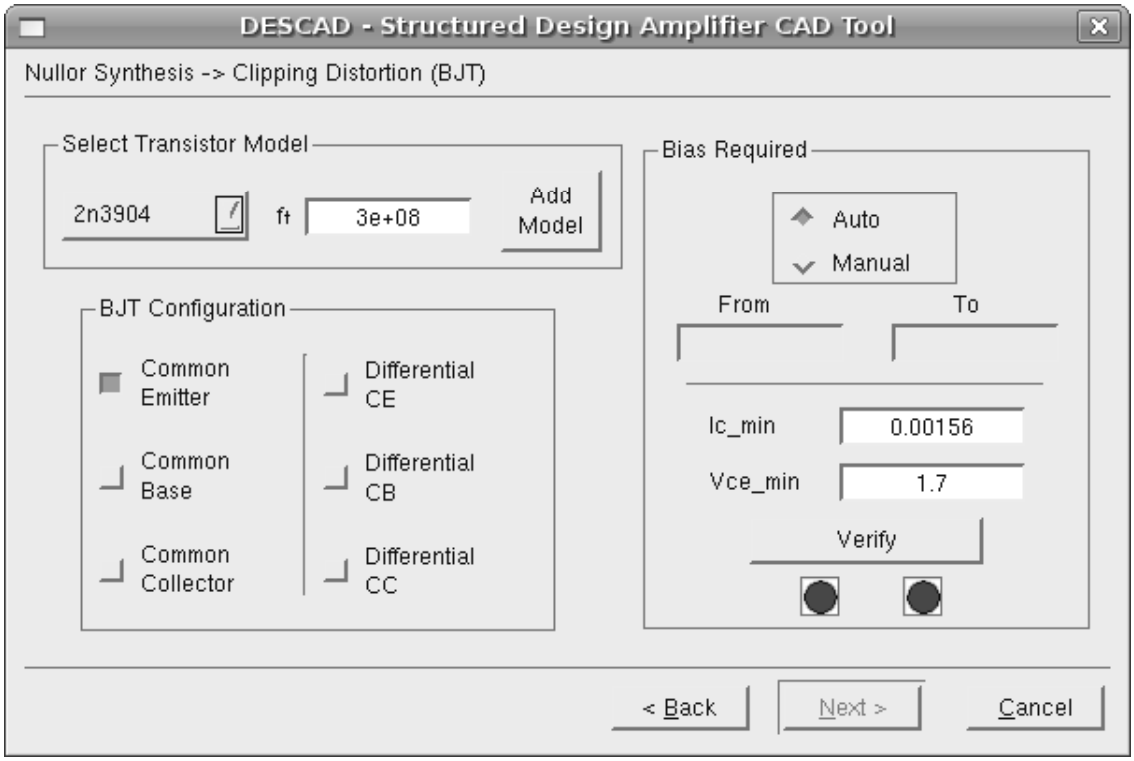

Figure 7.9: Minimum values in order to avoid clipping are calculated and shown to the user.

origin will be the highest.

From the results shown in Figure 7.11 it can be seen that the design does not reach the desired bandwidth. Therefore, the compensation process should be enforced. The tool informs this situation to the user by the message shown in Figure 7.12.

As stated in the information window, the first stage to be adjusted for the bandwidth behaviour is the noise stage. Again, the noise stage window appears and the user input the bias value within the range shown in the window as presented in Figure 7.13.

Once the bias value for the noise stage has been entered and the generated noise has been validated, the LP-product step is repeated. It can be seen in Figure 7.14 that the three poles found in the design are all dominant and the design process can continue. The final step, the Butterworth-like poles placement, shown in Figure 7.15 indicates where the phantom zero should be located in order to achieve the desired behavior. Selecting the option to place the compensation device at the input port the design tool provides the type and value for the element (Figure 7.16).

The last window in the wizard is a summary of the designed amplifier (Figure 7.17), here all the requirements are shown and the devices employed for the nullor synthesis are detailed as well in an easy to understand format. Once the user press the *Finish* button, the netlist of the circuit is generated. This netlist is in spice-like format and provides the small-signal elements of the active elements. The netlist is provided in Table 7.2.

The verification process is performed using the HSPICE verification software [14], frequency output performance is the desired Butterworth response as it can be seen in Figure 7.18. The poles and zeros of the system are shown in Table 7.3. Finally, the noise generated

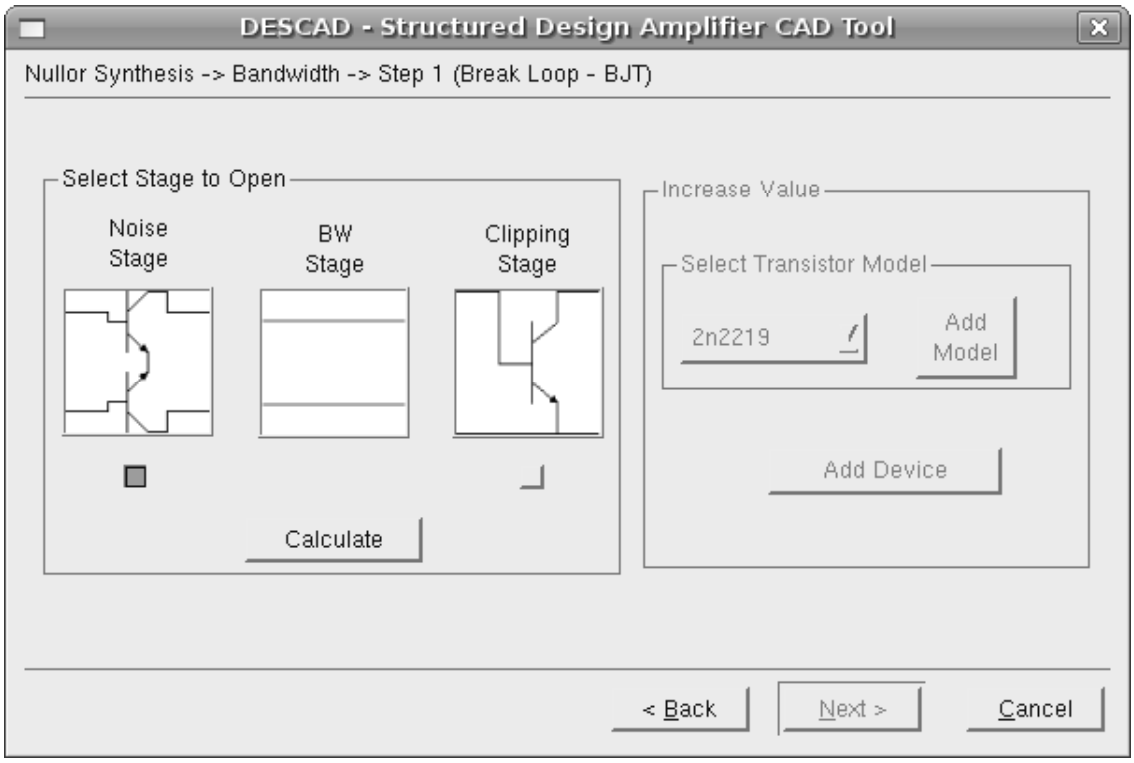

Figure 7.10: This window shows the option to select where the loop can be broken to perform the LP-product calculation.

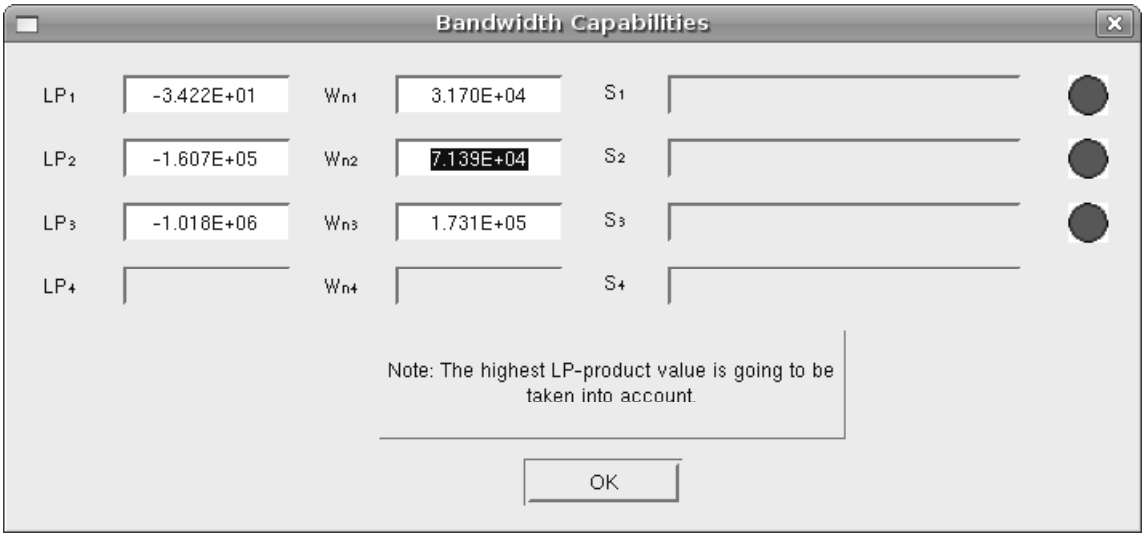

Figure 7.11: Here are shown the pole value, the LP-product result and if this is a dominant pole or not. These are arranged from the highest position to the lowest.

Figure 7.12: The tool informs the user that the already designed stages do not reach the desired bandwidth value.

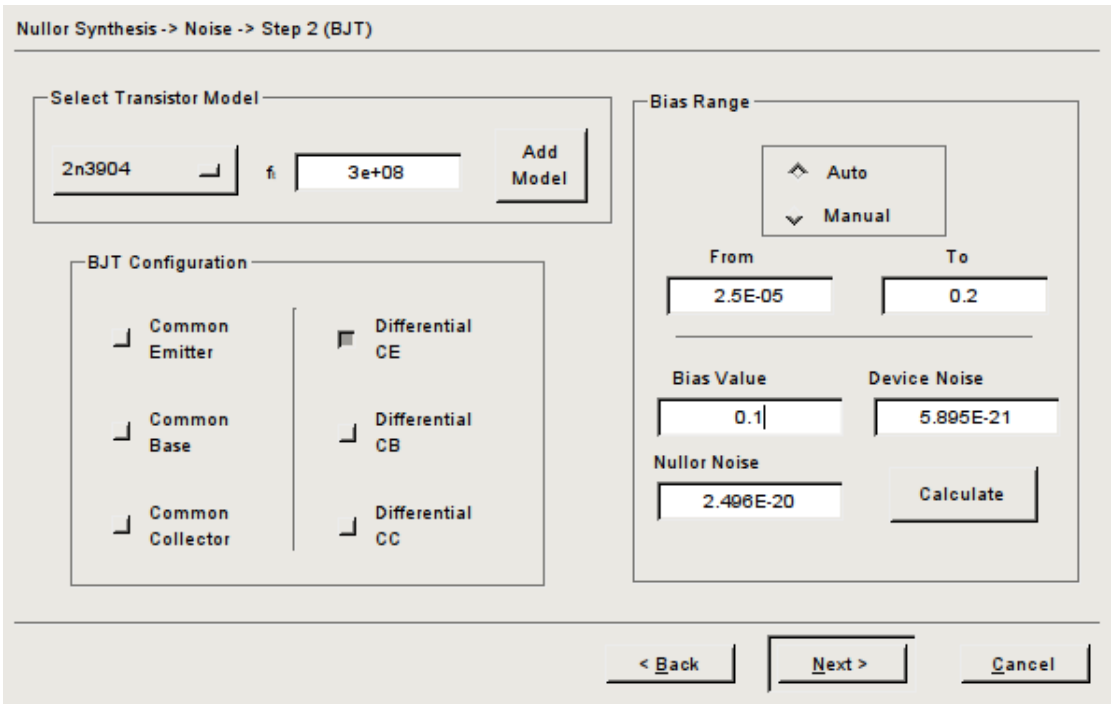

Figure 7.13: Bandwidth compensation by means of manual bias adjustment at the noise stage.

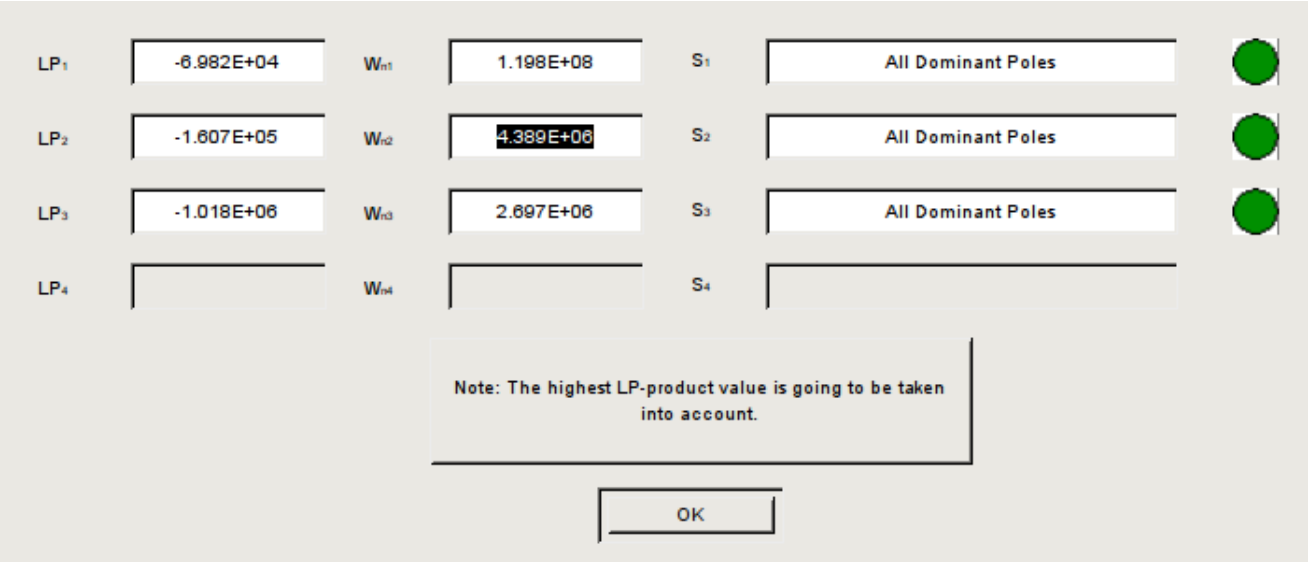

Figure 7.14: After the noise stage bias adjustment, there are three dominant poles.

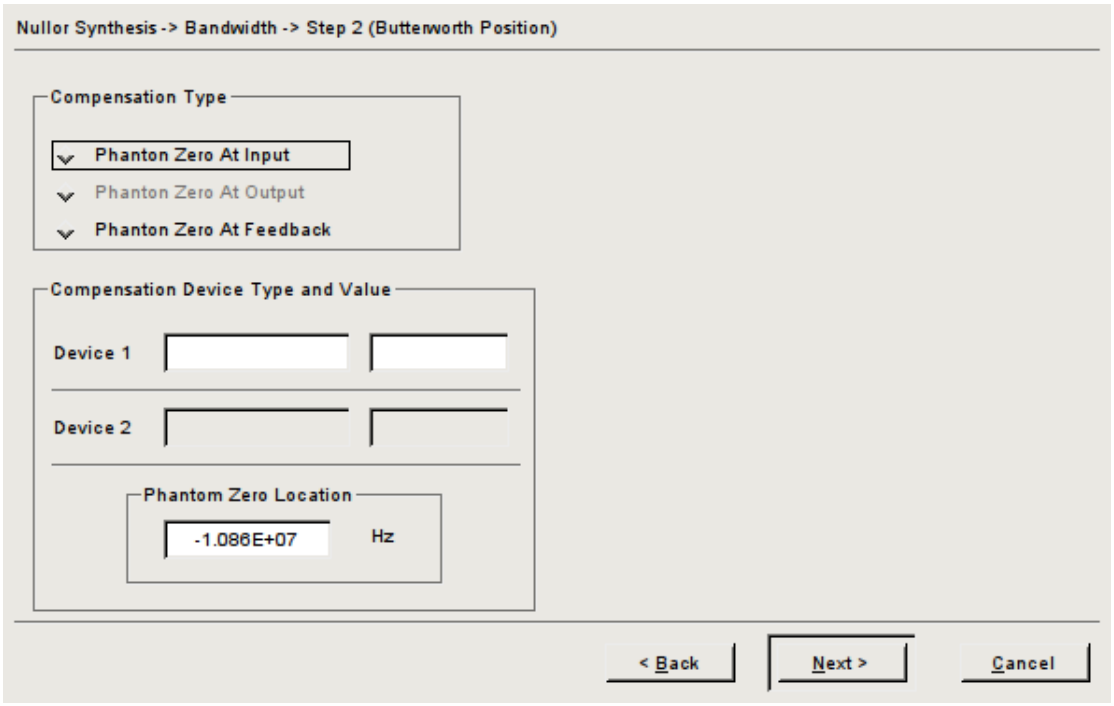

Figure 7.15: For a Butterworth-like behavior the tool shows where the phantom zero should be located.

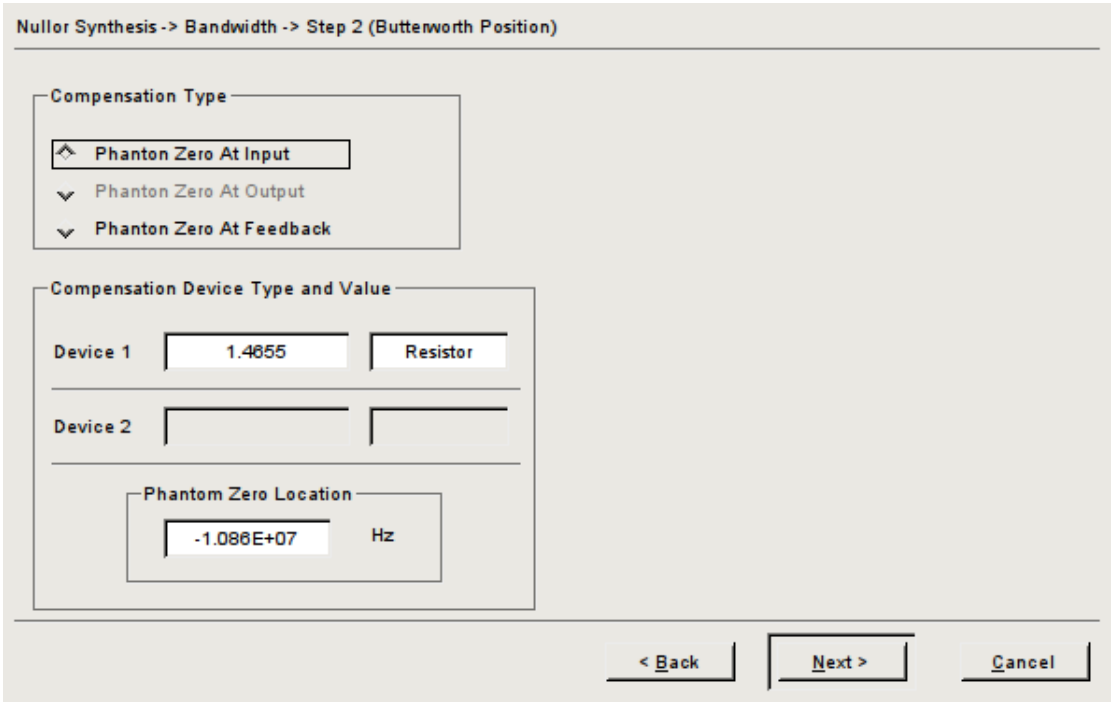

Figure 7.16: Selecting the compensation type to be a phantom zero placed at the input port, the DESCAD tool calculates the right type and value of the component.

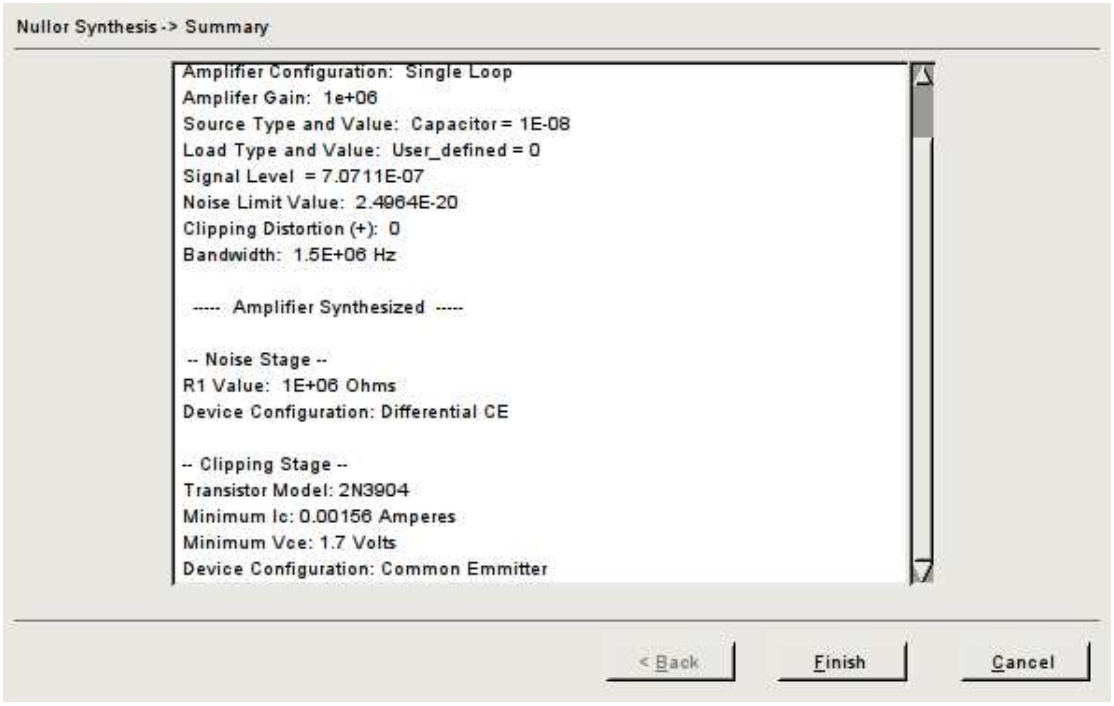

Figure 7.17: Final screen of the wizard, it contains a summary of the design.

```
 *This is the netlist for the designed amplifier
is 0 1 7.07107e-07
cs 1 0 1.000000E-08
cpi1 2 0 5.845188E-10
rpi1 2 0 2.153969E+02
gm1 4 0 2 0 1.933175E+00
cpi2 4 0 2.270322E-11
rpi2 4 0 6.887324E+03
gm2 0 3 4 0 6.045889E-02
rl 3 0 10e3
cl 3 0 100e-12
R1 3  2 1E+06
rph 1 2 1.465495E+00
```

Table 7.2: Generated netlist for the transimpedance amplifier in example 1.

is depicted in Table 7.4.

7.2 Low-Noise Single-loop Voltage Amplifier Test Case

A very interesting of study is the design of a low-noise amplifier. The low noise amplifier is a system where the noise level is carefully handled; it is usually the first block in a radio receiver that amplifies the signal from the antenna with as little distortion and additional noise as possible to be suitable for being processed by the first mixer [68].

It is necessary to establish that the design specs for a low-noise amplifier must be satisfied completely. The two most important aspects to be taken care of, among all, are the noise generated and the operating bandwidth.

The low-noise amplifier to be designed is a voltage amplifier (Figure 7.19) aimed to be employed in a radio receiver [69]. The design specs are as follows (Table 7.5):

Because the source impedance is modelled as a single resistance, it is possible to provide a noise spec in terms of noise figure. The preceding example provided two devices (one resistor in parallel with one capacitor) making it impossible to provide the desired noise in terms of the noise figure parameter.

The process to fulfill each screen for the basic specs is avoided given the fact that it was detailed in the previous example. Therefore, just the results for each synthesized stage is presented. These results are summarized in Table 7.6.

As for the final step of the design, the verification process takes place. Using the HSPICE [14] circuit simulation software the bandwidth of the amplifier is shown in Figure 7.20. Employing the same software it is possible to obtain where are the poles and zeros placed, these locations are presented in Table 7.7. Last, the noise values for the desired frequency (466 MHz) are shown in Table 7.8.

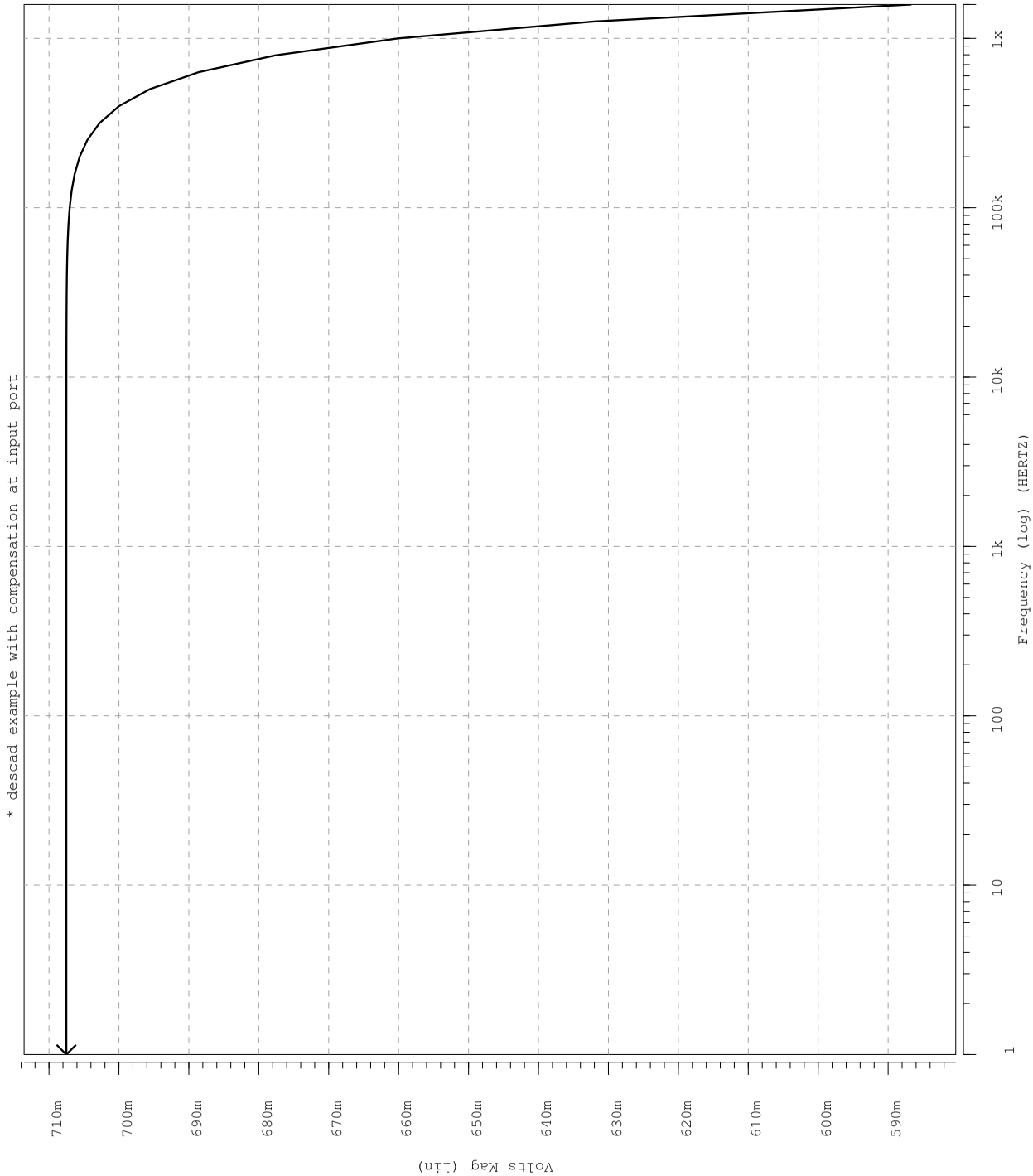

Figure 7.18: The bandwidth performance of the single-loop transimpedance amplifier shown in example 1.

```
****** HSPICE -Z-2007.03 32-BIT
******
 * descad example with compensation at input port
 ******   pole/zero analysis              tnom=  25.000 temp=  27.000
******

     input =  0:is                    output = v(3


     input =  0:is                    output = v(3

     poles (rad/sec)                  poles ( hertz)
*****************************************************************************
    real             imag             real             imag
  -11.7518x         13.0945x         -1.8704x          2.0841x
  -11.7518x        -13.0945x         -1.8704x         -2.0841x
   15.6080x          0.                2.4841x          0.
   -1.2431g          0.             -197.8429x          0.
 ***** no zeros found *****     coeffn =    6.0098m

 ***** constant factor =  6.009e+36      *****
```

Table 7.3: Poles and zeros of the transimpedance amplifier circuit in example 1.

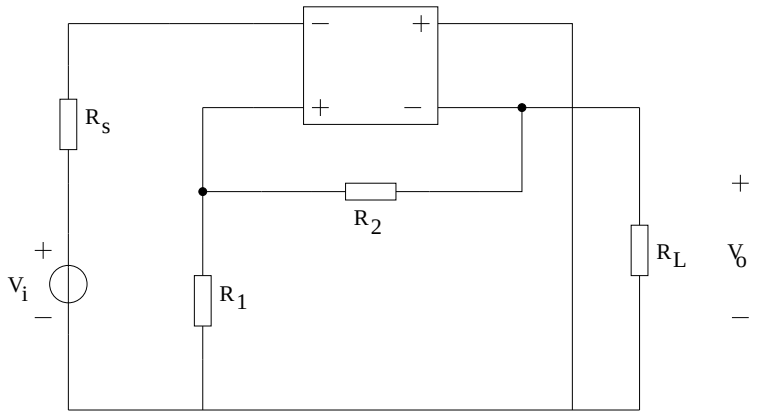

Figure 7.19: Nullor-base voltage amplifier.

```
****** HSPICE -Z-2007.03 32-BIT
******
 * descad example with compensation at input port
 ******  noise analysis                  tnom=  25.000 temp=  27.000
******

    frequency =   1.5000x      hz

**** resistor squared noise voltages (sq v/hz)

element     0:rph        0:rpi1       0:rpi2       0:rl         0:r1
      rs  155.1173p     56.3769p      4.6735f    5.888e-20     12.1472f
     1/f    0.           0.           0.           0.           0.
   total  155.1173p     56.3769p      4.6735f    5.888e-20     12.1472f
      rx  117.1077k    855.9209k     44.0665k    188.4737     856.0560k

**** total output noise voltage         = 211.5110p       sq v/hz
                                        =  14.5434u       v/rt hz
     transfer function value:
       v(3)/is                          = 847.8717k
     equivalent input noise at is
                                        =  17.1529p       /rt hz
```

Table 7.4: Noise behaviour of the transimpedance amplifier in example 1.

Input Level	0.025 Volts
Source Impedance	65 Ω
Output Level	0.175 Volts
Output Impedance	25 kΩ
Noise Figure	2 dB
Clipping Allowed	0 %
Bandwidth	466 MHz

Table 7.5: Design specifications.

Noise Stage	
Feedback Noise	1.847E-19 $\frac{V^2}{Hz}$
Source Noise	1.077E-18 $\frac{V^2}{Hz}$
Noise Stage Device (Single BFR520)	4.453E-19 $\frac{V^2}{Hz}$
Clipping Stage	
Clippin Stage Device (Differencial CE BFR520)	$I_{c_{min}} = 0.00579A$ $V_{CE} = 1.05V$
LP-Product Poles	
Two Dominant Poles	$P_1 = -5.098E7$ $P_2 = -1.374E9$
Compensation Device and Value	
Capacitor at the Feedback Network	$C_{ph} = 4.162E - 12$

Table 7.6: Summary of the results given by DESCAD for each synthesized stage.

```
1 ****** HSPICE -Z-2007.03 32-BIT  (Feb 28 2007)  pcnt
 ******
  *second phd example using hspice
  ******   pole/zero analysis              tnom=  25.000 temp=  25.000
 ******

      input =  0:vs                    output = v(4


      input =  0:vs                    output = v(4

      poles (rad/sec)                  poles ( hertz)
*****************************************************************************
    real             imag             real             imag
    8.8700g          0.               1.4117g          0.
  -20.8679g         11.7006g         -3.3212g          1.8622g
  -20.8679g        -11.7006g         -3.3212g         -1.8622g
      zeros (rad/sec)                    zeros ( hertz)
*****************************************************************************
    real             imag             real             imag
  -20.6489g          0.               -3.2864g         0.
  472.4842g          0.               75.1982g         0.
```

Table 7.7: Location of poles and zeros provided by the HSPICE software.

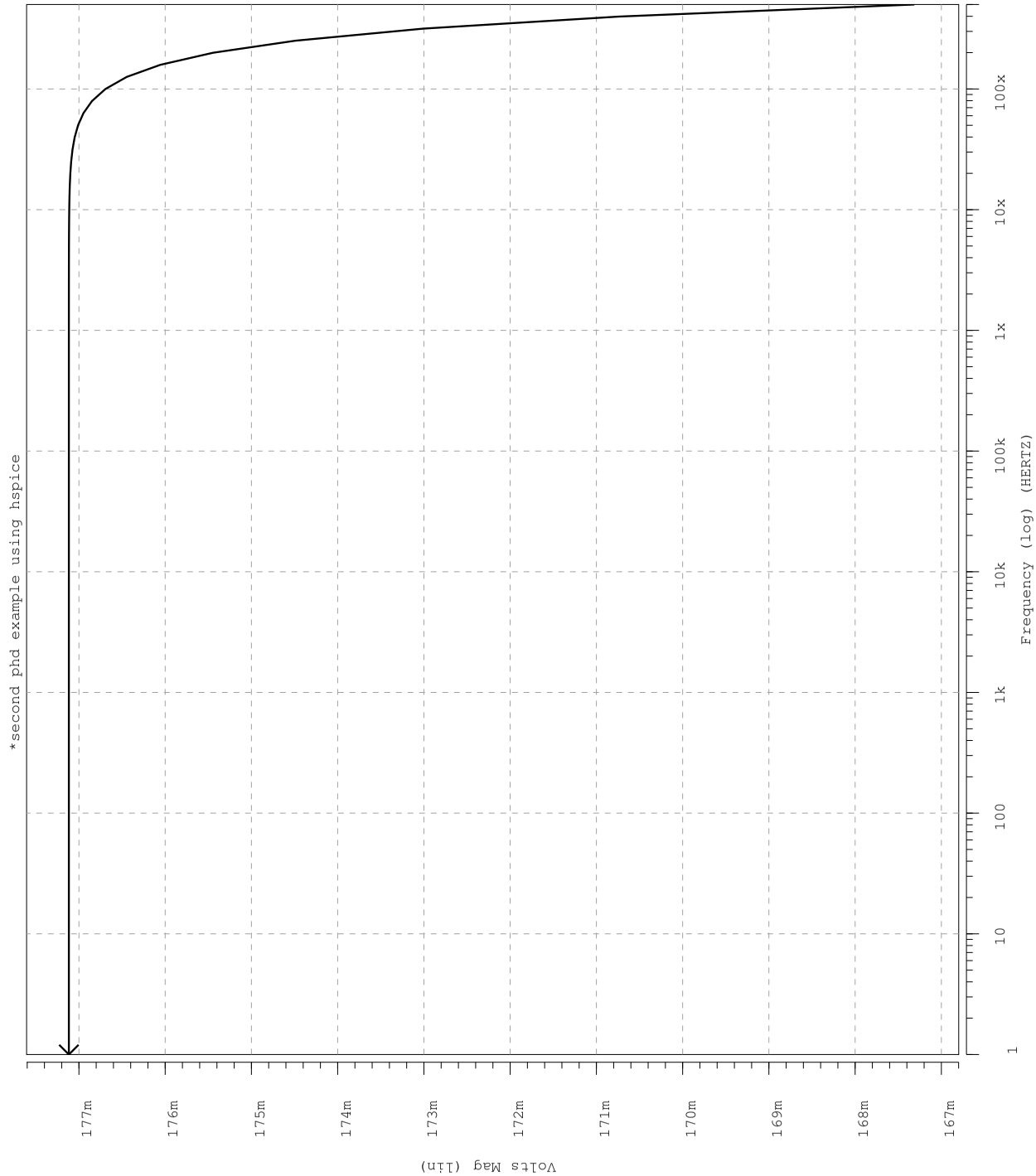

Figure 7.20: The bandwidth performance of the single-loop voltage amplifier shown in Example 2.

```
****** HSPICE -Z-2007.03 32-BIT  (Feb 28 2007) pcnt
******
 *second phd example using hspice
 ******  noise analysis                   tnom=  25.000 temp=  25.000
******

    frequency = 466.0000x      hz

**** resistor squared noise voltages (sq v/hz)

element     0:rs         0:rpi1       0:rpi2       0:rl         0:r1
      rs   48.6450a    5.832e-19    4.947e-19    6.792e-23      6.9652a
     1/f    0.           0.           0.           0.           0.
   total   48.6450a    5.832e-19    4.947e-19    6.792e-23      6.9652a
      rx  438.2185     511.9497     243.0980      10.1550      74.1571

element     0:r2
      rs    1.2838a
     1/f    0.
   total    1.2838a
      rx   77.9863

**** total output noise voltage         =  57.9719a       sq v/hz
                                        =   7.6139n       v/rt hz
     transfer function value:
       v(4)/vs                          =   6.7418
     equivalent input noise at vs
                                        =   1.1294n       /rt hz
```

Table 7.8: Noise values for the resistors in the voltage amplifier of the Example 2.

7.3 Single-loop Current Amplifier Test Case

The last example to be presented is the design of a single-loop current amplifier. Besides the main concern about the frequency performance of the circuit, another issue to be addressed is the noise contribution. This example aims to be as simple as possible but clear enough to show how the tool is able to handle this kind of task. The specs for the amplifier are given in Table 7.9.

Input Level	$1\mu Amps$
Source Impedance	2.5 $K\Omega$
Output Level	$10\mu Amps$
Output Impedance	12.5 $K\Omega$
Noise Level	10 dB
Clipping Allowed	0%
Bandwidth	1kHz

Table 7.9: Specs for the single-loop current amplifier.

Like the previous example the input impedance is comprised of a resistor, therefore the noise level can be provided in noise figure units. The results found for each synthesized stage are provided in Table 7.10.

Noise Stage	
Feedback Noise	2.717E-23 $\frac{A^2}{Hz}$
Source Noise	6.63E-24 $\frac{A^2}{Hz}$
Noise Stage Device (Differential CE 2N2219)	2.068E-23 $\frac{A^2}{Hz}$
Clipping Stage	
Clippin Stage Device (Single 2N2219)	$I_{c_{min}} = 30\mu A$ $V_{CE} = 0.701V$
LP-Product Poles	
Dominant Pole	$P_1 = -1.009E6$
Non-Dominant Pole	$P_2 = -5.195E6$
Compensation Device and Value	
Inductor at the Input Port	$L_{ph} = 263.66\mu H$

Table 7.10: Summary of the results given by DESCAD for each synthesized stage for the single-loop current amplifier.

The frequency response of the design, using HSPICE, is shown in Figure 7.21, the poles and zeros locations are provided in Table 7.11 and last, the noise behavior is given in Table 7.12.

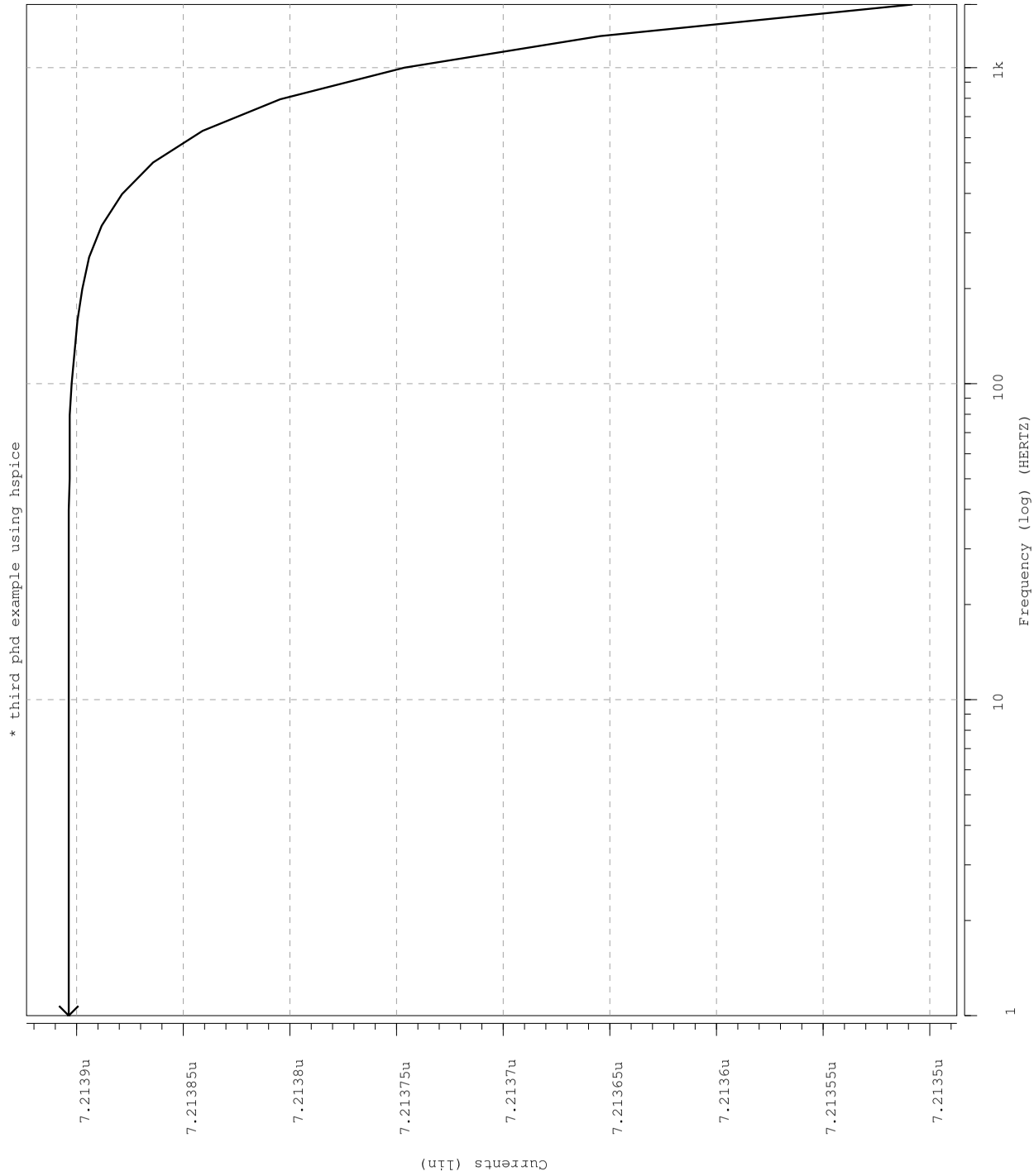

Figure 7.21: The bandwidth performance of the single-loop current amplifier shown in Example 3.

```
 ****** HSPICE -Z-2007.03 32-BIT  (Feb 28 2007)
 ******
  * third phd example using hspice
  ******   pole/zero analysis              tnom=  25.000 temp=  25.000
 ******

      input =  0:is                    output = i(rl


      input =  0:is                    output = i(rl

      poles (rad/sec)                  poles ( hertz)
*****************************************************************
    real            imag            real            imag
 -953.9842k         0.           -151.8313k         0.
  -12.1755x         0.             -1.9378x         0.
 -132.0355x         0.            -21.0141x         0.
 ***** no zeros found *****     coeffn =    7.8230u
```

Table 7.11: Location of poles and zeros for the Example 3 amplifier.

7.4 Conclusions

The numerical results provided by the tool depend on the spice parameters for each model employed on the design. Because of this, the results obtained will have various degrees of inaccuracy if compared to designs using parameters obtained by lab measurements. To clarify this situation the results obtained from the transimpedance amplifier explained in Example 1 will be compared to the results provided by [1]. The only difference between these two implementations is the fact that the feedback element employed in the implementation by [1] is a capacitor.

For both designs the active devices employed are the *2N3904* transistors; the optimal small-signal parameters for the input and output stages employed in [1] are given in Table 7.13. As for the values obtained using the development tool, they are shown in Table 7.14. Using the values found on the datasheet of the *2N3904* the computed results shows that, for the input stage, the adequate bias value is several orders smaller than the one measured in the lab. This bias value, in consequence, cause that the values for r_π, c_π and g_m vary greatly to the ones physically measured. The computed bias value for the output stage is about half the Verhoeven's design. Nevertheless, the r_π and c_π values are closer although the g_m value is also almost the half. This can only be explained for the fact that the computed results are based using parameters that were taken under ideal conditions.

Notice that the bias values for both implementations are on the same order but differ on the magnitude. These values are before the LP-product calculation. To perform the LP-product calculation for the design in [1] it is necessary to place a resistor in parallel to the

```
****** HSPICE -Z-2007.03 32-BIT  (Feb 28 2007)
******
 * third phd example using hspice
 ******  noise analysis                    tnom=  25.000 temp=  25.000
******

    frequency =   1.0000k      hz

**** resistor squared noise voltages (sq v/hz)

element     0:rs         0:rpi1       0:rpi2       0:rl         0:r1
      rs  1.714e-22    9.115e-25    4.583e-22      0.         7.023e-23
     1/f    0.           0.           0.           0.           0.
   total  1.714e-22    9.115e-25    4.583e-22      0.         7.023e-23
      rx    5.1009       5.1009      48.9808       3.6396a    510.0889m

element     0:r2
      rs  6.321e-22
     1/f    0.
   total  6.321e-22
      rx    4.5908

**** total output noise current         = 1.333e-21       sq amp/hz
                                        =  36.5083p       amp/rt hz
     transfer function value:
       i(rl)/is                         =   5.1009
     equivalent input noise at is
                                        =   7.1572p       /rt hz
```

Table 7.12: Noise values at the output and input ports for the current amplifier of the Example 3.

Small-signal parameter	Input stage 2N3904 @ Ic=90 μA	Output stage 2N3904 @ Ic=3.6 mA
r_π [kΩ]	52	1.2
c_π [pF]	7	46
g_m [mA/V]	3.5	132

Table 7.13: The small-signal parameters for the input and output stages provided in the example found in [1].

Small-signal parameter	Input stage 2N3904 @ Ic=25 μA	Output stage 2N3904 @ Ic=1.56 mA
r_π [kΩ]	861.6	1.65
c_π [pF]	2.392	51.5
g_m [mA/V]	0.483	60.45

Table 7.14: The small-signal parameters for the input and output stages provided by the DESCAD design tool.

feedback capacitor. For the DESCAD design it is not necessary since the feedback element is a resistor. The poles for the design in [1] are presented in Table 7.15, and the poles for the DESCAD design are in Table 7.16. The difference for the number of poles is because there is a simplification for the original loop gain in [1]. The poles for the system are given by

$$L(s) = \frac{-\beta_{f1}\beta_{f2}R_L}{[1 + sr_{\pi 2}c_{\pi 2}][R_L + R_f + sR_L(R_fC_f + R_fC_L + r_{\pi 1}C_L]}$$

assuming that $C_f >> C_L$ and $R_f >> r_{\pi 1}$, the previous expression becomes

$$L(s) = \frac{-\beta_{f1}\beta_{f2}R_L}{R_f} \cdot \frac{1}{[1 + sr_{\pi 2}c_{\pi 2}][1 + sR_LC_f]}$$

because of this simplification the DESCAD design shows one more pole. Neither the DESCAD software nor the KMNA maple library employ a simplification process. Therefore, this is a disadvantage because the LP-product will be affected and so the calculated attainable bandwidth.

Pole Number	**Position**
p_1	$-15.9kHz$
p_2	$-2.9MHz$

Table 7.15: Two poles are found for the design by Verhoeven after simplification.

Pole Number	**Position**
p_1	$-34.22Hz$
p_2	$-160.7kHz$
p_3	$-1.018MHz$

Table 7.16: All three poles of the circuit without simplification.

Once the compensation process has been enforced, the small-signal AC simulation outputs for both designs are shown in Figure 7.22 and Figure 7.18 respectively. The most noticeable difference is the shape of the curves. For the design from [1] not only the cutoff frequency is several orders lower than expected but the output level is not acceptable also; it seems that the feedback capacitor has an important influence on the overall performance of the amplifier despite the fact that the circuit poles are close to the desired Butterworth position.

On the other hand, the DESCAD design performs right on the bandwidth range and the output level even surpases the desired expected level according to the simulation results.

It has been seen that the small-signal values for the devices employed on the numerical calculations do have significant influence on the overall results for the amplifier performance. The small-signal values can be obtained from several manufacturing vendors but this information is not entirely realiable since it is provided just as a reference and not as a definitive source of information.

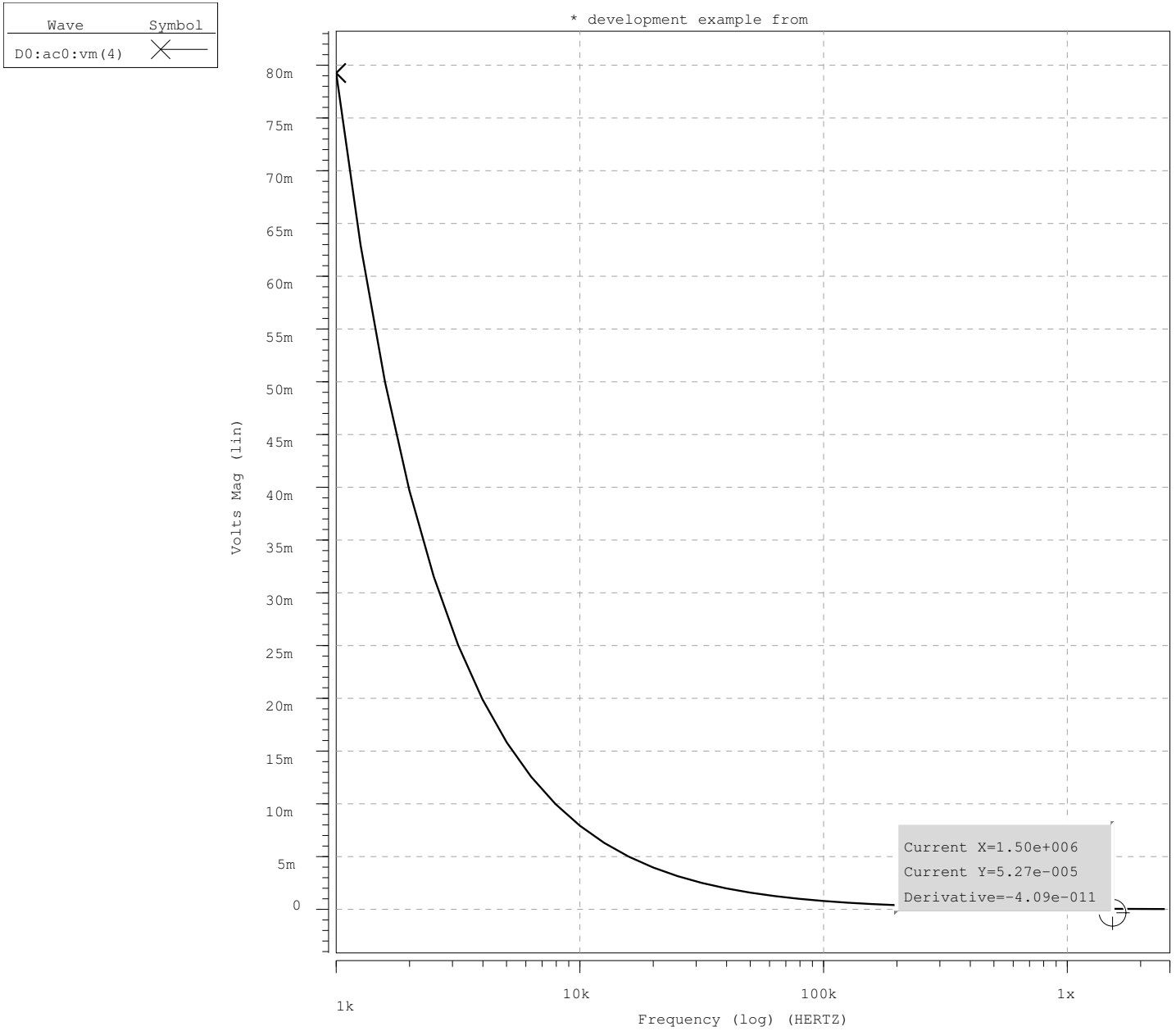

Figure 7.22: Output signal as generated by HSPICE for the Verhoeven's design example.

Chapter 8

Conclusions

Although the methodology of Structured Design is not the only existing one for achieving analog circuit design, it is the only one having a simplified scheme based on Circuit Theory and Electronics fundamentals. It represents an attemp to diminish the dependence on the experience of the designer during the design flow, and as a result it allows novice designers to understand and gain insight of more complex design challenges with less effort and training time. The Structured Design methodology starts by assuming an ideal solution, which, in its ideality, satisfies all requirements. The design methodology is in fact the path that brings us from the start line (ideal solution) to the goal (real feasible solution) following well defined guidelines.

This thesis has been focused on developing a CAD frame for the design automation of negative-feedback amplifiers under the guidelines established by the Structured Design methodology.

In this document not only the concepts of Structured Design were clarified but also brought these concepts from merely statements, guidelines and assumptions to lines of code that properly coded in the developed CAD frame leads the designer through the problem of design from the initial specs to the final design.

Regarding the design aspects, orthogonality of the design variables (noise, distortion and bandwidth) has been respected as far as possible, but a short link between the noise and bandwidth has been kept with the idea of providing a degree of flexibility to the user. During the design of the bandwidth stage, on the contrary, flexibility has been reduced in order to avoid exploiting due to an unnecessary increase of the order of the system.

The CAD frame (DESCAD) follows a wizard approach that helps the designer along the whole design flow. This has been achieved with cost of code length but it gives more insight to the designer. DESCAD, in its current version, has been totally programmed in C++ (for the numerical back-end) resorting to a friendly graphic user interface. This programming language was chosen because its object oriented structure that allowed to translate as faith as possible the concepts of the Structured Design methodology. The DESCAD numerical engine also employs the well-proof numerical analysis routines with a connection to a small symbolic analysis engine developed under MAPLE.

The wizard approach of DESCAD proved to be useful because its step-by-step basis during the first stages of design for new users. This has been achieved by generating an easy-to-operate tool where the learning curve could be kept as flat as possible. Doubtless is

this a result of a very well oriented graphical user interface.

The final design is a small-signal amplifier that can be verified by using a standard verification tool. In fact, DESCAD is able to generate the input file for SPICE-like simulators. After the verification using simulation tools, the next step would be to implement the design and this way validate how accurate is the design tool. The differences obtained by the implementation will help to identify, add or correct the parameters to increase the complexity needed to move away from the ideal solution and to be closer to the particular, and desired, solution.

Future Work

It can be seen that the ultimate goal to translate the Structured Design methodology into a CAD frame has been accomplished but there is still some work pending in order to make it a robust design framework. It is necessary to include the MOS devices into the CAD frame. At this moment only the noise calculations have been finished, the implementation for the two remaining stages should be trivial using the BJT devices as a guide. Another improvement would be to modify the bandwidth algorithm in order to handle in an efficient way systems with more than two poles as it is now.

A very interesting option to be added into the DESCAD frame is the process implemented by [70] that approach the design process providing the values for the small-signal devices of each stage using genetic and evolutionary algorithms. This method is already developed in the C language so its adaptation to the C++ language should be simple.

Last, in order to keep this CAD frame project active the most viable option is to release the code into the open source community. This ambitious approach could involve the INAOE community and contributors, either amateur or professional, to share thoughts and expertise for the improvement of the DESCAD tool. Guided by the CAD group this project could be divided into various sub-projects in a research and development approach with the ultimate goal to create an industrial class development framework.

Appendix A

Noise Source Movement

A.1 Single-loop Topologies

These are the noise source movements for the following three single-loop amplifier topologies:

- Transadmittance Amplifier.
- Transimpedance Amplifier.
- Current Amplifier.

A.1.1 Transadmittance Amplifier

This analysis does not take into account the external source value but its impedance value.

Step 1. Noise sources are added to the circuit. Noise current i_n is placed between the positive and negative input ports of the nullor. Noise voltage source u_n is placed at the input branch of the nullor. Figure A.1.

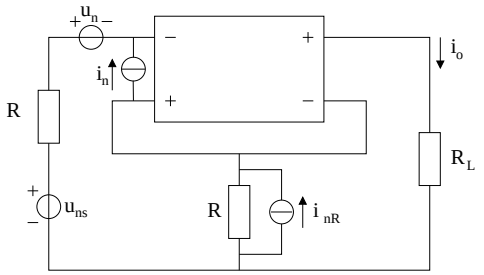

Figure A.1: Nullor based transadmittance amplifier.

Step 2. Add noise sources u_{ns} and $-u_n$. It becomes $u'_{ns} = u_{ns} - u_n$. Figure A.2.

Step 3. I-shift on i_n. Figure A.3.

Step 4. Norton-Thevenin transformation on i_n. Reducing the sources: $u''_{ns} = u_{ns} - u_n + R_S i_n$ and $i'_{nR} = i_{nR} - i_n$. Figure A.4.

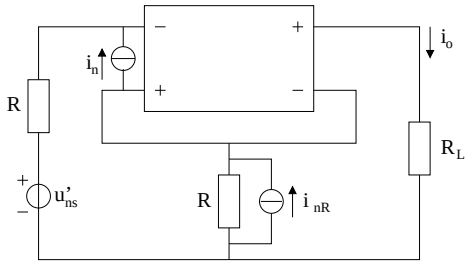

Figure A.2: Nullor based transadmittance amplifier. Step 2.

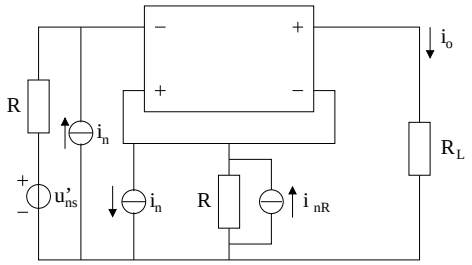

Figure A.3: Nullor based transadmittance amplifier. Step 3.

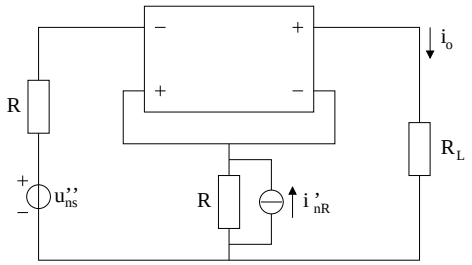

Figure A.4: Nullor based transadmittance amplifier. Step 4.

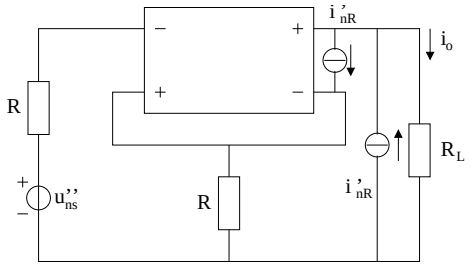

Figure A.5: Nullor based transadmittance amplifier. Step 5.

Step 5. I-shift on i'_{nR}. Figure A.5.

Step 6. The noise current source placed at the output port of the nullor is disregarded because it has no effect on the overall noise behaviour. Only the noise current source placed in parallel to the load resistance is taken into account. Figure A.6

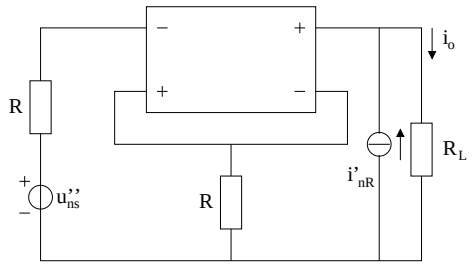

Figure A.6: Nullor based transadmittance amplifier. Step 6.

Step 7. Two-port transformation on i'_{nR}. The only Chain-matrix parameter to be taken into account is B. Figure A.7.

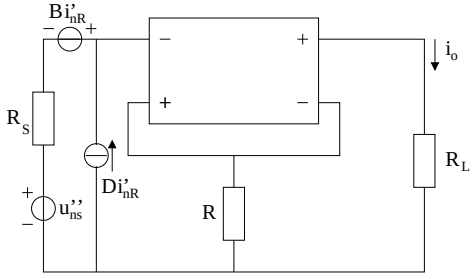

Figure A.7: Nullor based transadmittance amplifier. Step 7.

A.1.2 Transimpedance Amplifier

This analysis does not take into account the external source value but its impedance value.

Step 1. Noise sources are added to the circuit. Noise current i_n is placed between the positive and negative input ports of the amplifier. Noise voltage source u_n is placed at the negative input branch of the nullor. Figure A.8.

Step 2. Noise sources i_{ns} and i_n are added. The result is $i'_{ns} = i_{ns} + i_n$. Figure A.9.

Step 3. V-shift is applied to u_n. Figure A.10.

Step 4. Norton-Thevenin transformation is performed on i_{n1}. The new noise voltage source is given by $u_{n1} = i_{n1} * R_1$. Figure A.11.

Step 5. The $-u_n$ source located at the input branch of the amplifier is converted to a noise current source by a Norton-Thevenin transformation. Noise source u_n and u_{n1} are added then they become $u'_{n1} = u_n + u_{n1}$. Figure A.12.

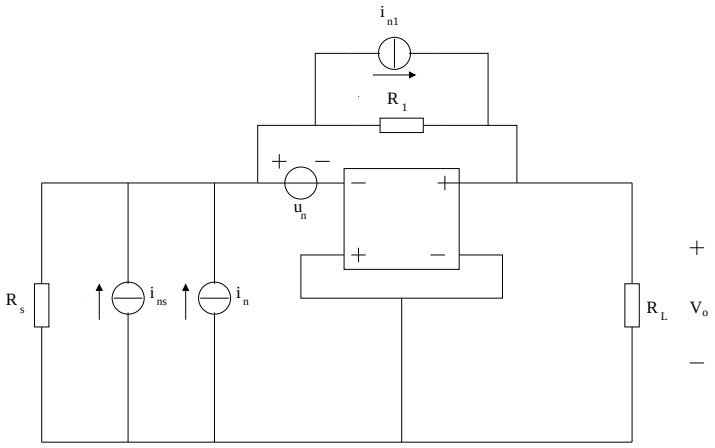

Figure A.8: Nullor based transimpedance amplifier.

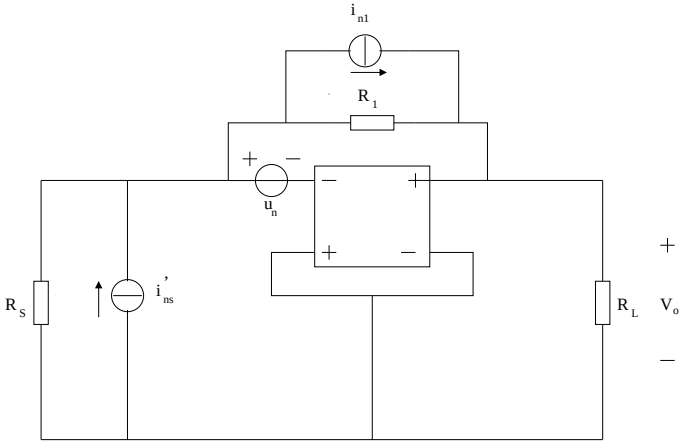

Figure A.9: Nullor based transimpedance amplifier. Step 2.

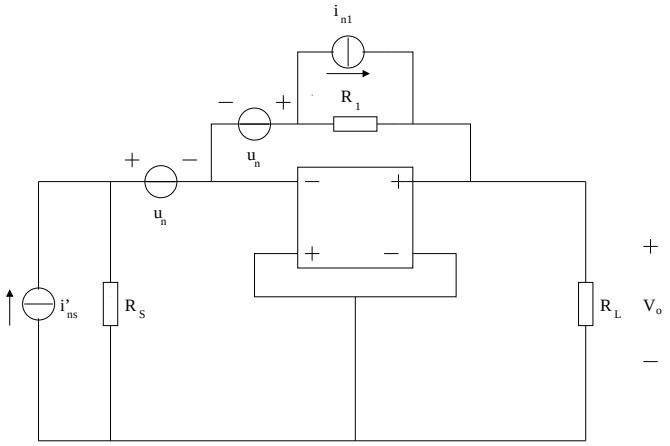

Figure A.10: Nullor based transimpedance amplifier. Step 3.

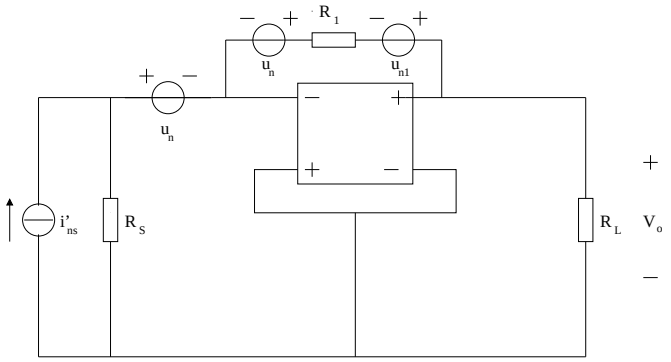

Figure A.11: Nullor based transimpedance amplifier. Step 4.

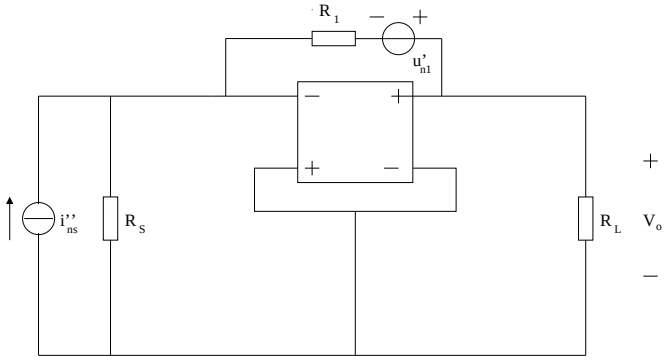

Figure A.12: Nullor based transimpedance amplifier. Step 5.

Step 6. V-shift transformation on u'_{n1} lets place this source at the output branch of the amplifier. While the other is located at the positive output port of the nullor, therefore this one can be not taken into account. Figure A.13.

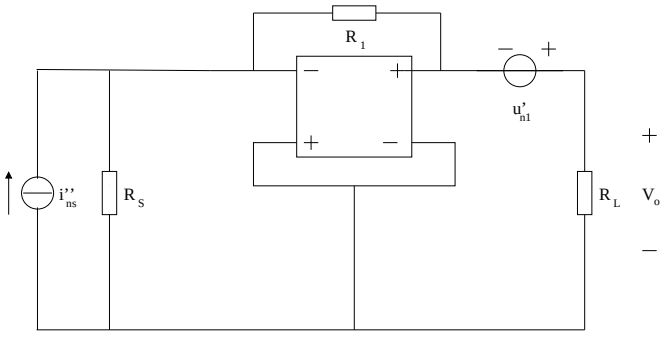

Figure A.13: Nullor based transimpedance amplifier. Step 6.

Step 7. Two-port transformation on u'_{n1}. The Chain-matrix parameter different of zero is C. Figure A.14.

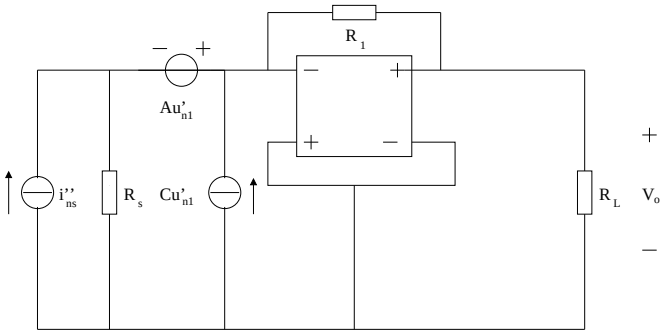

Figure A.14: Nullor based transimpedance amplifier. Step 7.

A.1.3 Current Amplifier

This analysis does not take into account the external source value but its impedance value.

Step 1. Noise sources are added to the circuit. Noise current i_n is placed between the positive and negative input ports of the amplifier. Noise voltage source u_n is placed at the positive input branch of the nullor. Figure A.15.

Step 2. Noise sources i_n and i_{ns} are simplified to a single source by $i'_{ns} = i_{ns} + i_n$. Figure A.16.

Step 3. V-shift is employed on u_n. Figure A.17.

Step 4. Norton-Thevenin transform on the $-u_n$ placed at the input branch converts this source to $i_{n'} = -\frac{u_n}{R_S}$. Then this source is added to i'_{ns} to become $i''_{ns} = i'_{ns} + i_{n'}$. The other source u_n is added to u_{n1} which gives as result $u'_{n1} = u_{n1} + u_n$. Figure A.18.

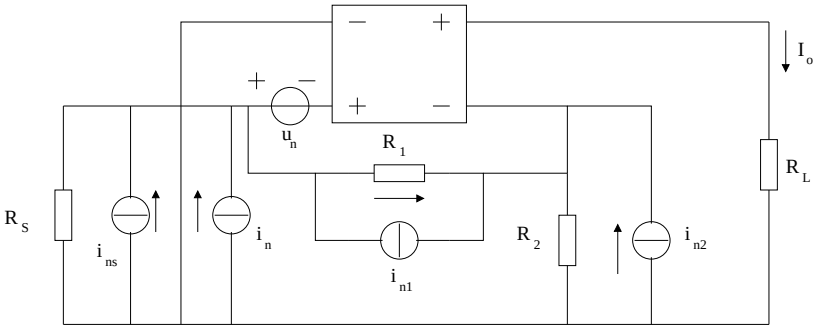

Figure A.15: Nullor based current amplifier.

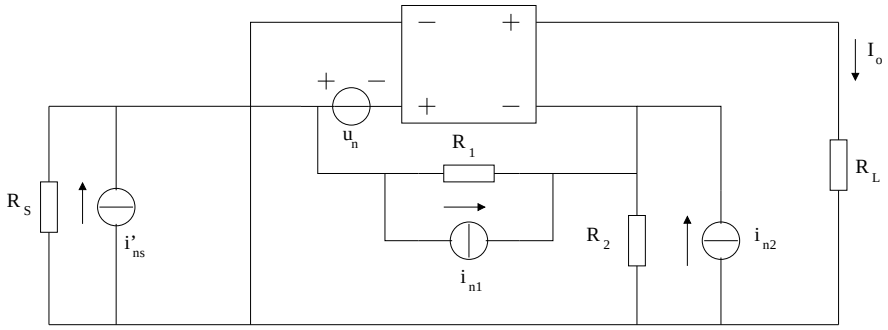

Figure A.16: Nullor based current amplifier. Step 2.

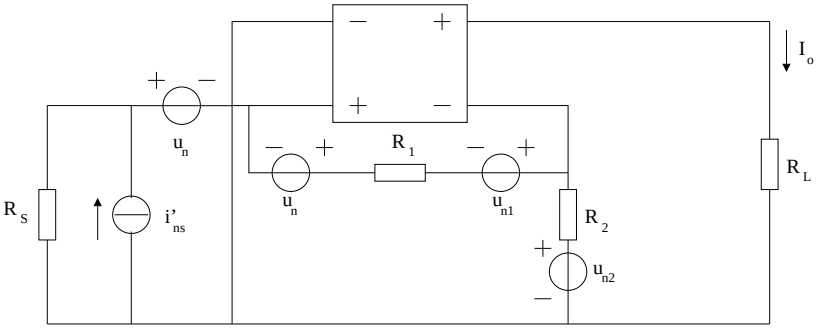

Figure A.17: Nullor based transimpedance amplifier. Step 3.

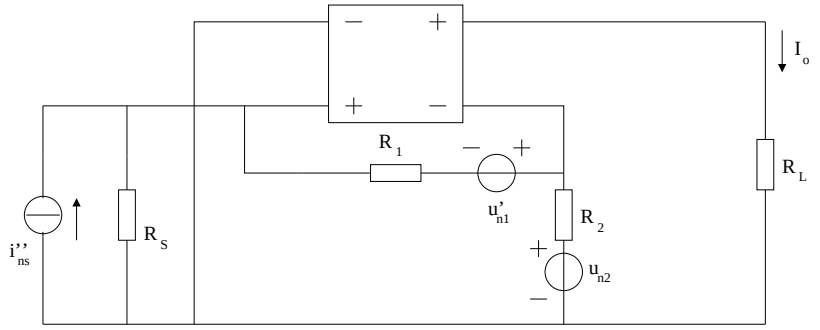

Figure A.18: Nullor based transimpedance amplifier. Step 4.

Step 5. V-shift is enforced to u'_{n1}. The first one is located at the negative output port of the nullor, again this source is not taken into account because properties of the nullor. The other one is added to u_{n2} and becomes $u'_{n2} = u_{n2} + u'_{n1}$. Figure A.19.

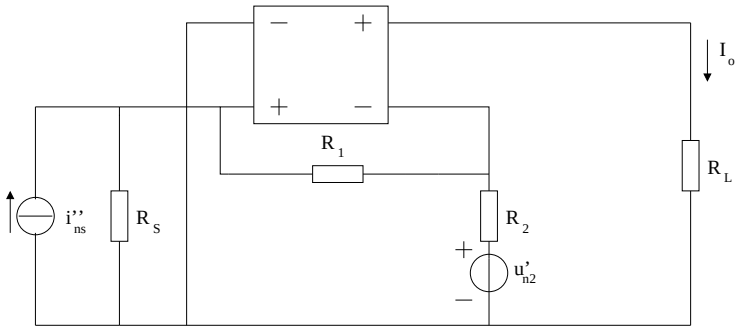

Figure A.19: Nullor based transimpedance amplifier. Step 5.

Step 6. Norton-Thevenin transformation on u'_{n2} turns it to i'_{n2}. I-shift is performed on this source. One of the noise sources is placed at the output port of the nullor, therefore it is discarded by properties of the nullor. The only source useful is the one located at the output port of the amplifier. Figure A.20.

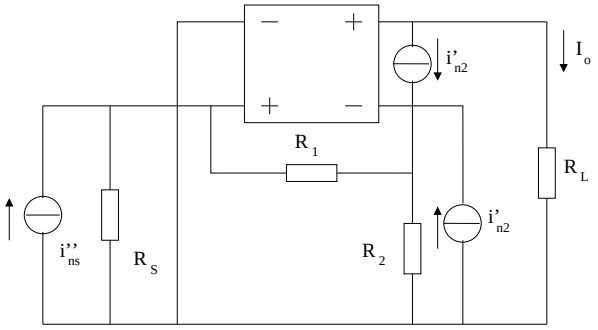

Figure A.20: Nullor based transimpedance amplifier. Step 6.

Step 7. The final step is to apply Two-port transformation on i'_{n2}. In terms of the Chain-matrix for this configuration, the only useful parameter is D. Figure A.21.

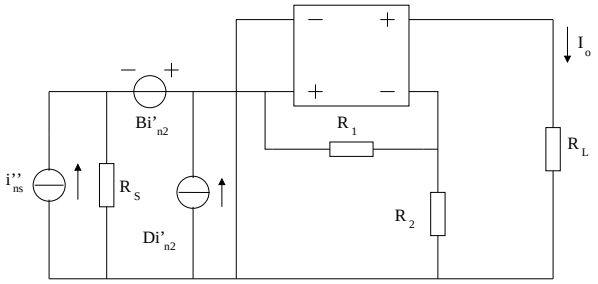

Figure A.21: Nullor based transimpedance amplifier. Step 7.

A.2 Double-loop Topology

Noise source movements are performed on the following topologies:

- Two-loop (B) Voltage Amplifier.
- Two-loop (B) Transadmittance Amplifier.
- Two-loop (B) Transimpedance Amplifier.
- Two-loop (B) Current Amplifier.

As with the previous single loop topologies, all of the following analyses do not take into account the external source value but their source impedance value.

A.2.1 Two-loop (B) Voltage Amplifier

Step 1. Noise sources are added to the circuit. Noise current i_n is placed between the positive and negative input ports of the nullor. Noise voltage source u_n is placed at the input branch of the amplifier. Figure A.22.

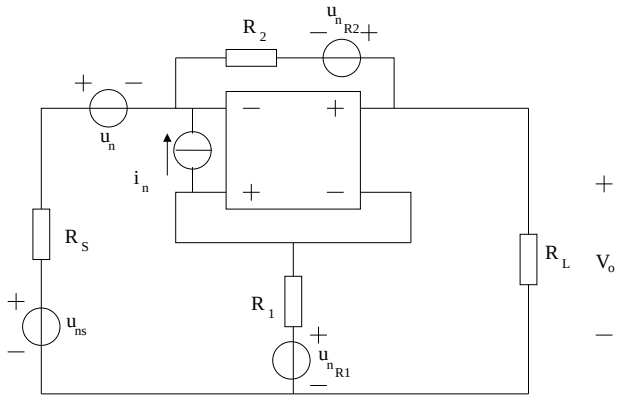

Figure A.22: Nullor based two-loop (B) amplifier. Voltage to voltage.

Step 2. Sources u_{ns} and $-u_n$ add into a single source $u'_{ns} = u_{ns} - u_n$. Figure A.23.

Step 3. I-shift is applied to i_n. Figure A.24

Step 4. By means of Norton-Thevenin transformation the two noise current sources created by I-shift are converted to voltage noise sources. Then the resulting sources are: $u''_{ns} = u'_{ns} + i_n R_s$ and $u'_{n_{R_1}} = u_{n_{R_1}} - i_n R_1$. Figure A.25.

Step 5. V-shift allows to move the source $u_{n_{R_2}}$. This shift creates two sources, one is located at the positive output port of the nullor and by properties of this device the noise source does not have any influence on the overall noise behaviour. The other noise source is placed at the output branch of the amplifier. Figure A.26.

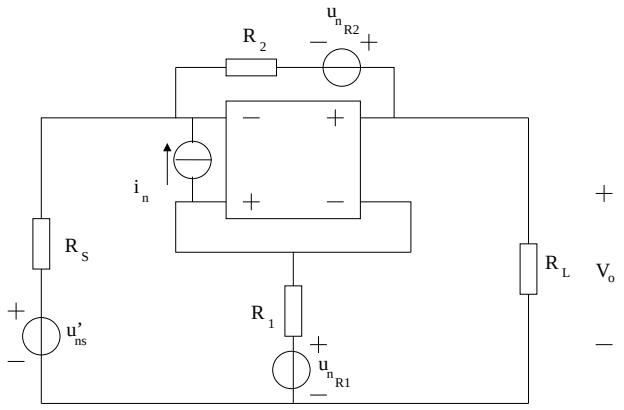

Figure A.23: Nullor based two-loop (B) amplifier. Voltage to voltage. Step 2.

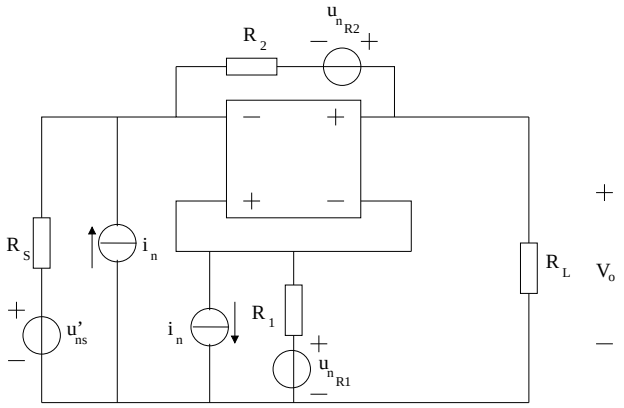

Figure A.24: Nullor based two-loop (B) amplifier. Voltage to voltage. Step 3.

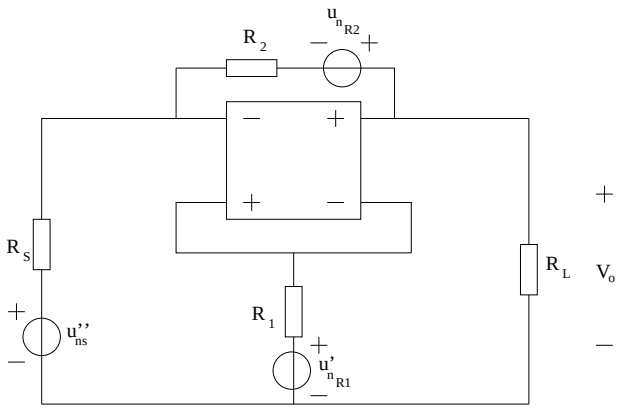

Figure A.25: Nullor based two-loop (B) amplifier. Voltage to voltage. Step 4.

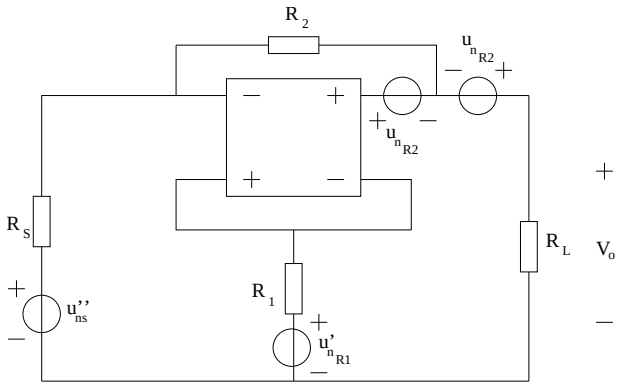

Figure A.26: Nullor based two-loop (B) amplifier. Voltage to voltage. Step 5.

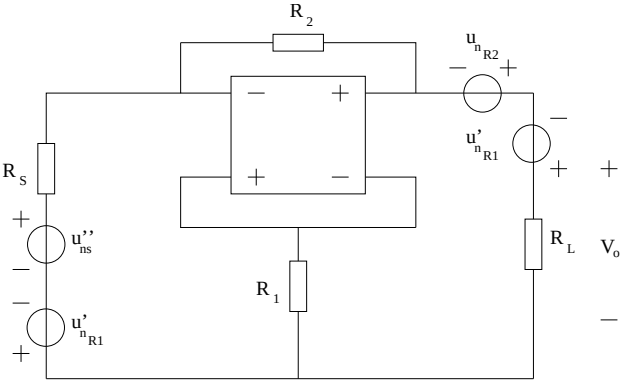

Figure A.27: Nullor based two-loop (B) amplifier. Voltage to voltage. Step 6.

Step 6. The noise voltage source $u'_{n_{R_1}}$ is shifted by the V-shift transformation. The two resulting sources are located as follows: one is placed in series to u'_{n_s} and the other is in series to $u_{n_{R_2}}$. Figure A.27.

Step 7. The noise sources are reduced into single sources. For the sources at the input: $u'''_{ns} = u''_{ns} - u'_{n_{R_1}}$, as for the sources at the output: $u'_{n_{R_2}} = u_{n_{R_2}} + u'_{n_{R_1}}$. Figure A.28.

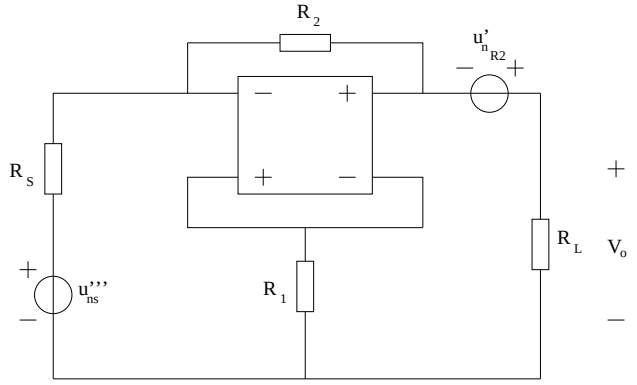

Figure A.28: Nullor based two-loop (B) amplifier. Voltage to voltage. Step 7.

Step 8. Two-port transformation is performed on $u'_{n_{R_2}}$. For this case both Chain-matrix parameters A and C are accounted. Figure A.29.

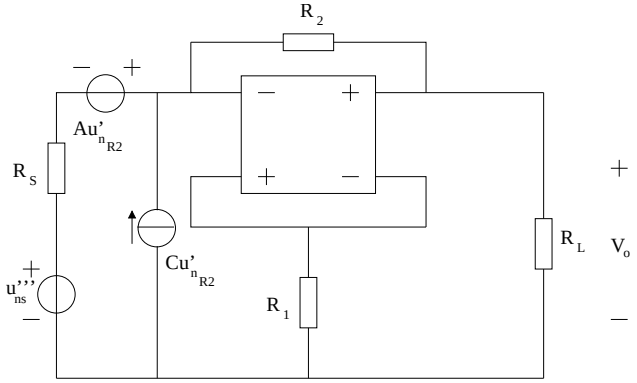

Figure A.29: Nullor based two-loop (B) amplifier. Voltage to voltage. Step 8.

Step 9. Finally the two noise sources created by the Two-port transformation are reduced. Noise voltage source $Au'_{n_{R_2}}$ is added directly to u'''_{ns}. Next is to apply Norton-Thevenin to $Cu'_{n_{R_2}}$ and convert it to a noise voltage source $\frac{Cu'_{n_{R_2}}}{R_S}$. This newly created source then is added to u'''_{ns}, therefore the circuit is noise free and the total noise contribution is represented as just one noise voltage source. Figure A.30.

A.2.2 Two-loop (B) Transadmittance Amplifier

Step 1. Noise sources are added to the circuit. Noise current i_n is placed between the positive and negative input ports of the nullor. Noise voltage source u_n is placed at

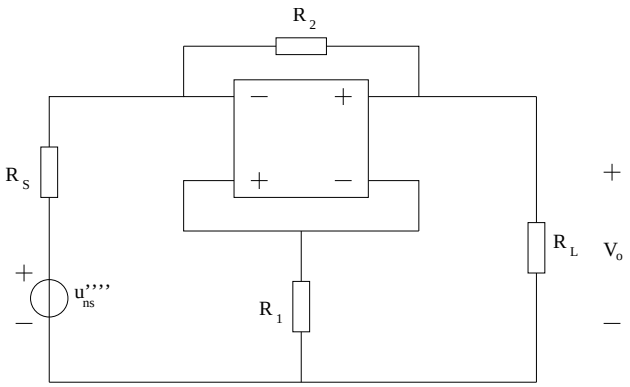

Figure A.30: Nullor based two-loop (B) amplifier. Voltage to voltage. Step 9.

the input branch of the amplifier. Figure A.31.

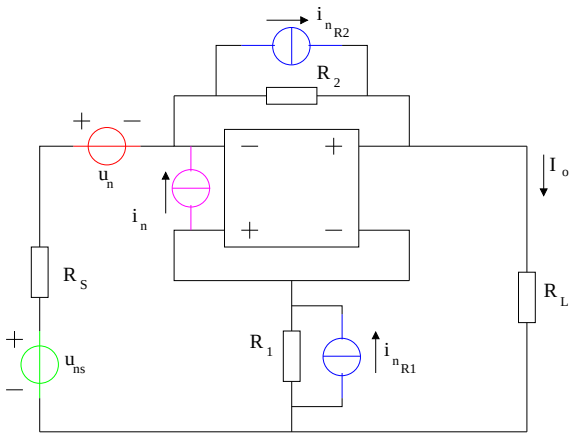

Figure A.31: Nullor based two-loop (B) amplifier. Voltage to Current.

Step 2. u_{ns} is in series with $-u_n$, then sources are simplified. The new source is $u'_{ns} = u_{ns} - u_n$. Figure A.32.

Step 3. By I-shift transformation i_n is placed at the input of the amplifier and in parallel to $i_{n_{R_1}}$. Figure A.33.

Step 4. Both i_n sources are reduced. For the source located at the input port, the Thevenin-Norton transformation converts it to $u_{n'} = i_n R_s$ then is added to u'_{ns}. The new reduced source is $u''_{ns} = u'_{ns} + u_{n'}$. Now, the parallel current sources are reduced to $i'_{n_{R_1}} = i_{n_{R_1}} - i_n$. Figure A.34.

Step 5. For the $i_{n_{R_2}}$ source an I-shift transformation is performed. One source is placed at the input port of the amplifier while the other is located at the output port of the amplifier. Figure A.35.

Step 6. Norton-Thevenin transformation on the source at the input port of the amplifier to convert it in a voltage source $u_{n''}$ then this source is reduced. The new source is $u'''_{ns} = u''_{ns} - u_{n''}$. Figure A.36.

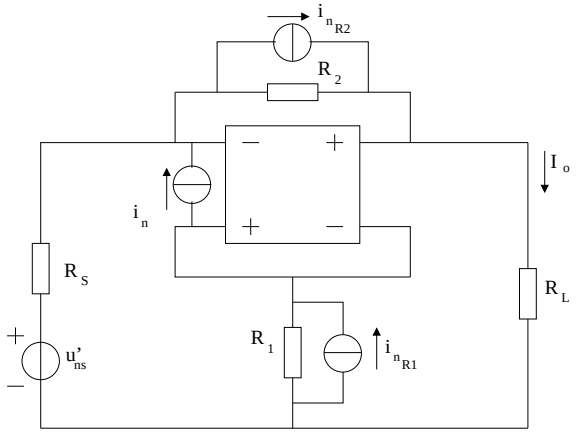

Figure A.32: Nullor based two-loop (B) amplifier. Voltage to Current. Step 2.

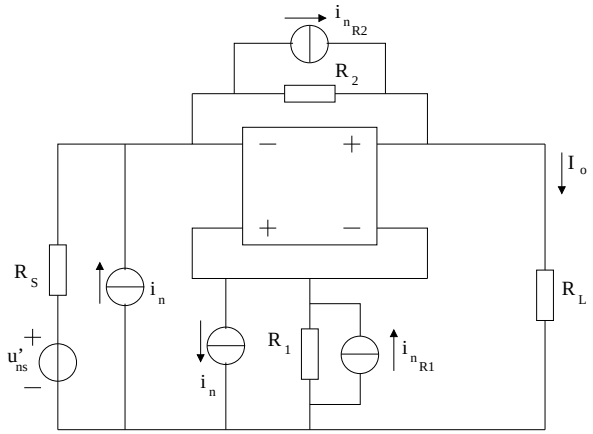

Figure A.33: Nullor based two-loop (B) amplifier. Voltage to Current. Step 3.

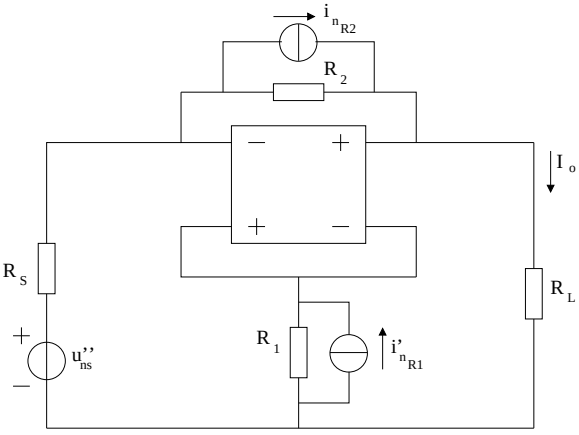

Figure A.34: Nullor based two-loop (B) amplifier. Voltage to Current. Step 4.

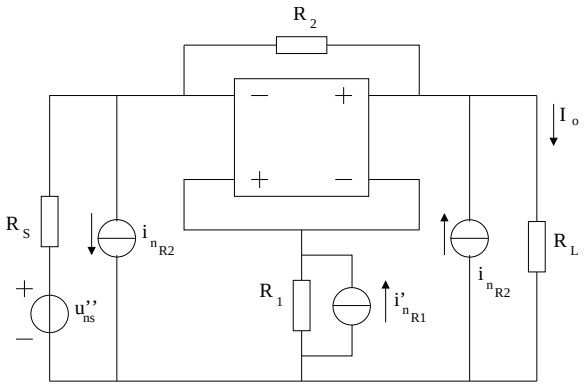

Figure A.35: Nullor based two-loop (B) amplifier. Voltage to Current. Step 5.

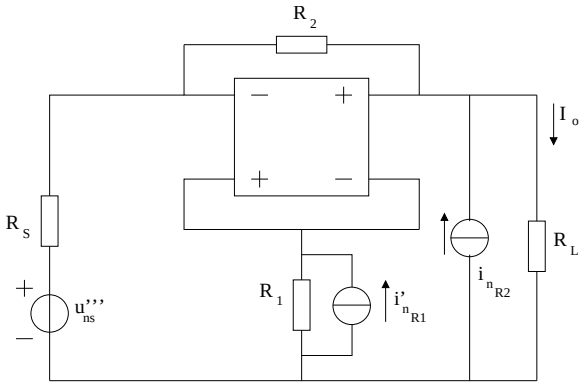

Figure A.36: Nullor based two-loop (B) amplifier. Voltage to Current. Step 6.

Step 7. I-shift for the $i'_{n_{R_1}}$ source place one source at the output port of the nullor and at the port of the amplifier. The source at the output port of the nullor is discarded. At the output port of the amplifier are two noise current sources. Figure A.37.

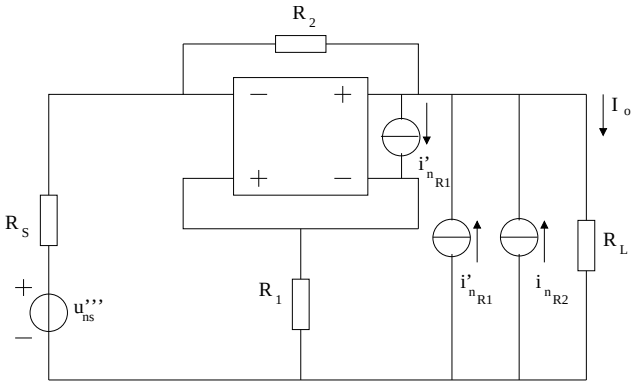

Figure A.37: Nullor based two-loop (B) amplifier. Voltage to Current. Step 7.

Step 8. The parallel noise current sources are reduced to become $i''_{n_{R_1}} = i'_{n_{R_1}} + i_{n_{R_2}}$. Figure A.38.

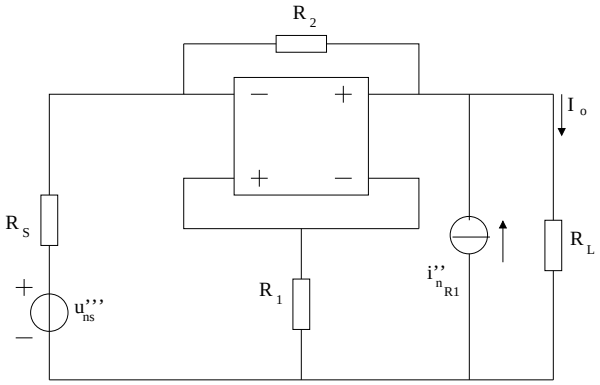

Figure A.38: Nullor based two-loop (B) amplifier. Voltage to Current. Step 8.

Step 9. Now, the source $i''_{n_{R_1}}$ is moved to the input port via Two-port transformation. The Chain-matrix parameters B and D must be taken into account. Figure A.39.

Step 10. This is the last step. The noise voltage source $Bi''_{n_{R_1}}$ is added directly to u'''_{ns} and the noise current source $Di''_{n_{R_1}}$ must be transformed to a voltage source, this is done by a Norton-Thevenin transformation. Once it becomes a voltage source is added directly to u'''_{ns} and noise sources are out of the amplifier and comprised on only one equivalent source. Figure A.40.

A.2.3 Two-loop (B) Transimpedance Amplifier

Step 1. Noise sources are added to the circuit. Noise current i_n is placed between the positive and negative input ports of the amplifier. Noise voltage source u_n is placed at the negative input branch of the nullor. Figure A.41.

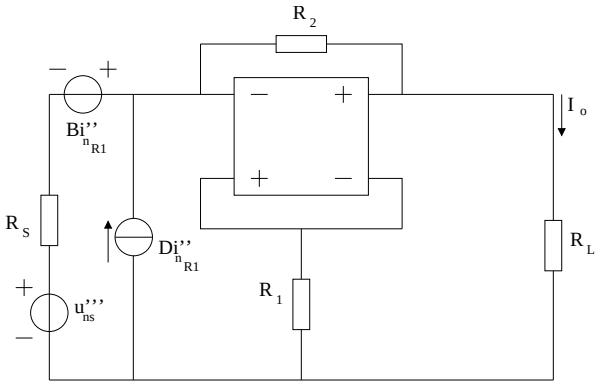

Figure A.39: Nullor based two-loop (B) amplifier. Voltage to Current. Step 9.

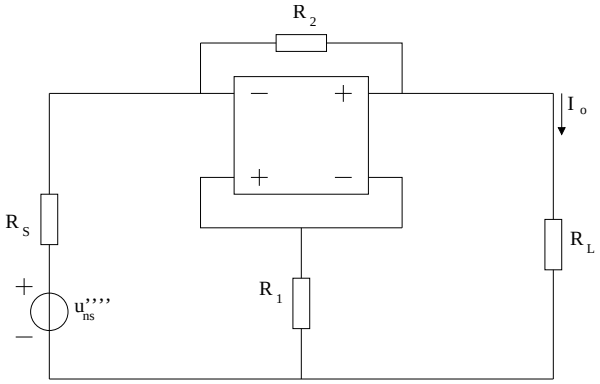

Figure A.40: Nullor based two-loop (B) amplifier. Voltage to Current. Step 10.

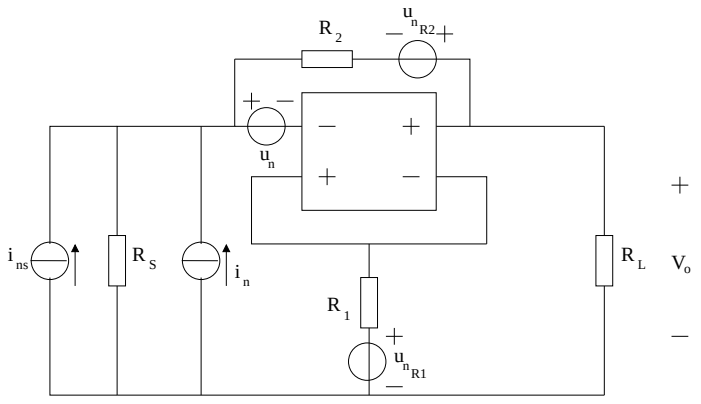

Figure A.41: Nullor based two-loop (B) amplifier. Current to Voltage.

Step 2. The noise current source i_{ns} and i_n are in parallel, these two sources are reduced into a single noise current source $i'_{ns} = i_{ns} + i_n$. Figure A.42.

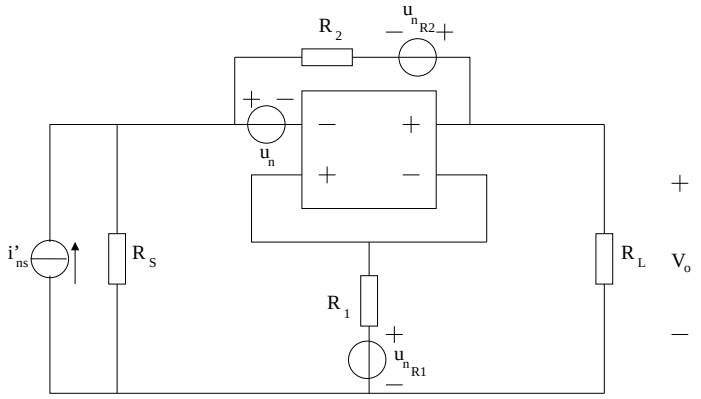

Figure A.42: Nullor based two-loop (B) amplifier. Current to Voltage. Step 2.

Step 3. V-shift is performed on u_n and is moved to the input branch of the amplifier and in series with $u_{n_{R_2}}$. Figure A.43.

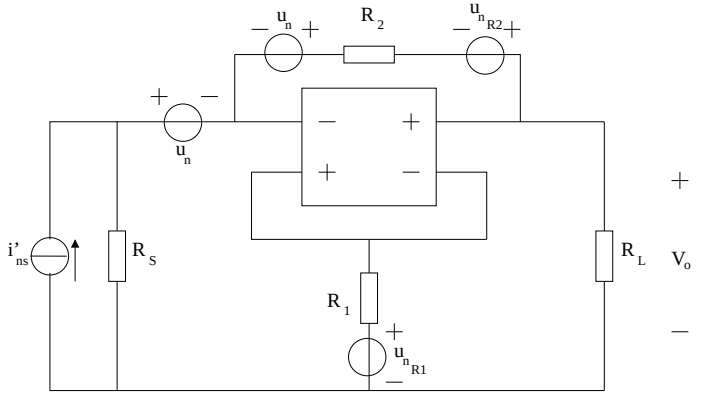

Figure A.43: Nullor based two-loop (B) amplifier. Current to Voltage. Step 3.

Step 4. For the source placed at the input branch the Norton-Thevenin transformation is applied in order to convert it into a noise current source $-u_n/R_S$. The other noise voltage source is reduced and becomes $u'_{n_{R_2}} = u_{n_{R_2}} + u_n$. Figure A.44.

Step 5. Since i'_{ns} is in parallel with $-u_n/R_S$ a reduction into a single source is performed giving as result $i''_{ns} = i'_{ns} - u_n/R_S$. For the source $u'_{n_{R_2}}$, V-shift is performed and is taken to the output port of the nullor and to the output branch of the amplifier. The nullor has no noise to add therefore the source placed at its output port is left aside. Figure A.45.

Step 6. The voltage noise source $u_{n_{R_1}}$ is moved by V-shift. One source is placed at the input branch of the amplifier and the other is in series with $u'_{n_{R_2}}$. Figure A.46.

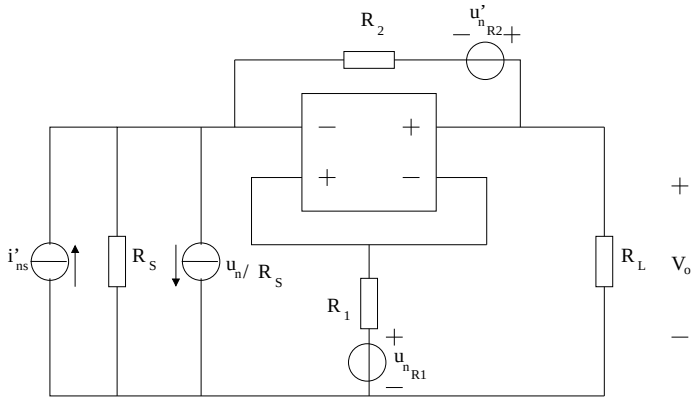

Figure A.44: Nullor based two-loop (B) amplifier. Current to Voltage. Step 4.

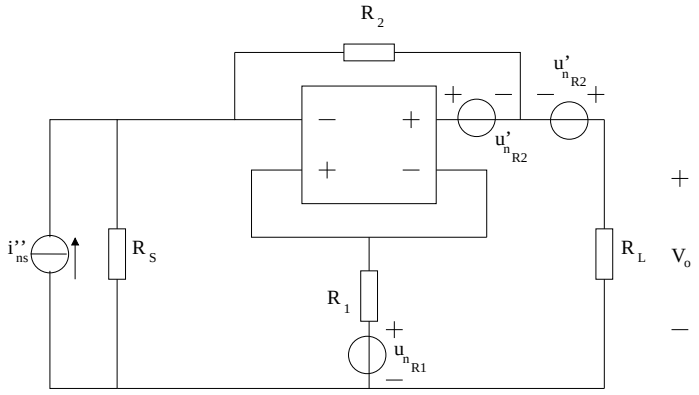

Figure A.45: Nullor based two-loop (B) amplifier. Current to Voltage. Step 5.

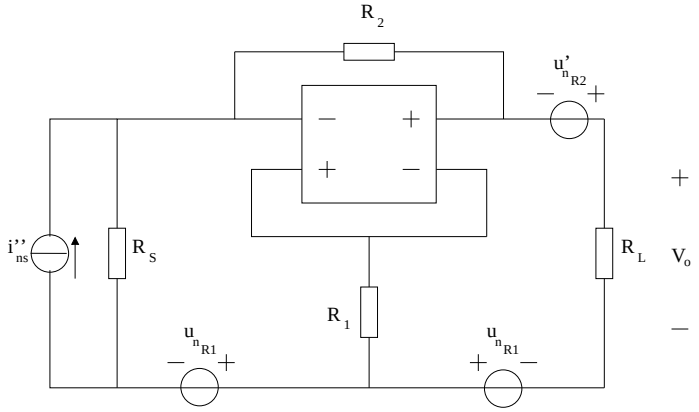

Figure A.46: Nullor based two-loop (B) amplifier. Current to Voltage. Step 6.

Step 7. A Thevenin-Norton transformation is applied to the noise voltage source at the input branch to become $-u_{n_{R_1}}/R_S$. As for the series sources at the output port, these are added giving as result $u''_{n_{R_2}} = u'_{n_{R_2}} + u_{n_{R_1}}$. Figure A.47.

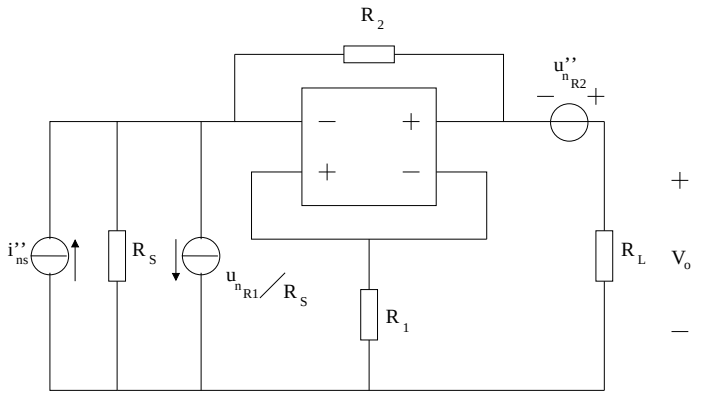

Figure A.47: Nullor based two-loop (B) amplifier. Current to Voltage. Step 7.

Step 8. The parallel current sources are reduced into $u'''_{ns} = u''_{ns} - u_{n_{R_1}}/R_S$. The voltage source at the output branch is taken to the input port by a Two-port transformation. Figure A.48.

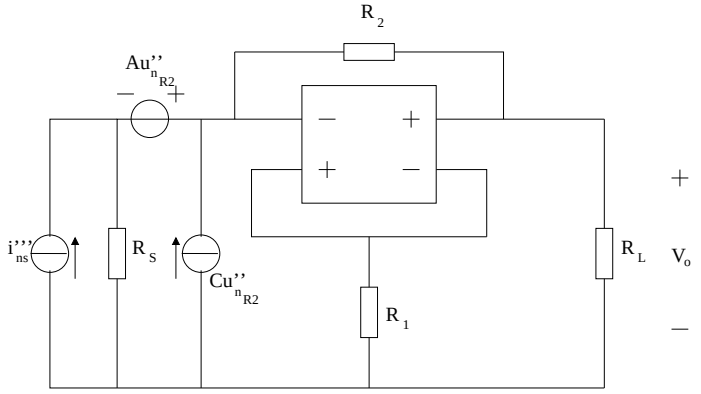

Figure A.48: Nullor based two-loop (B) amplifier. Current to Voltage. Step 8.

Step 9. First the parallel noise current sources u'''_{ns} and $Cu''_{n_{R_2}}$ are reduced. Then the source $Au''_{n_{R_2}}$ is converted by Norton-Thevenin and placed in parallel to the reduced noise current source. Figure A.49.

Step 10. Finally the parallel sources are added and the amplifier is represented by only one current noise source located outside the amplifier. Figure A.50.

A.2.4 Two-loop (B) Current Amplifier

Step 1. Noise sources are added to the circuit. Noise current i_n is placed between the positive and negative input ports of the amplifier. Noise voltage source u_n is placed at the negative input branch of the nullor. Figure A.51.

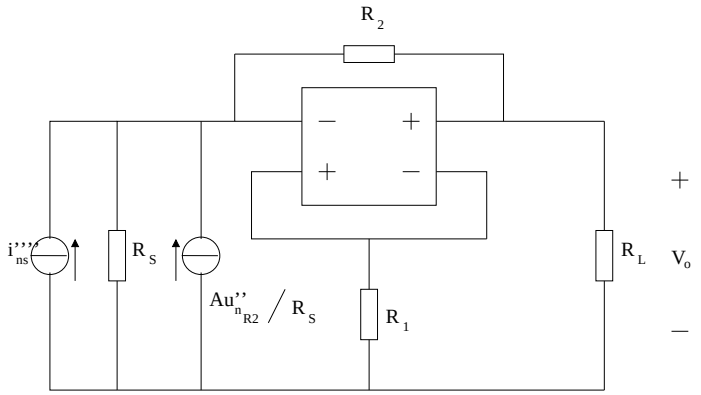

Figure A.49: Nullor based two-loop (B) amplifier. Current to Voltage. Step 9.

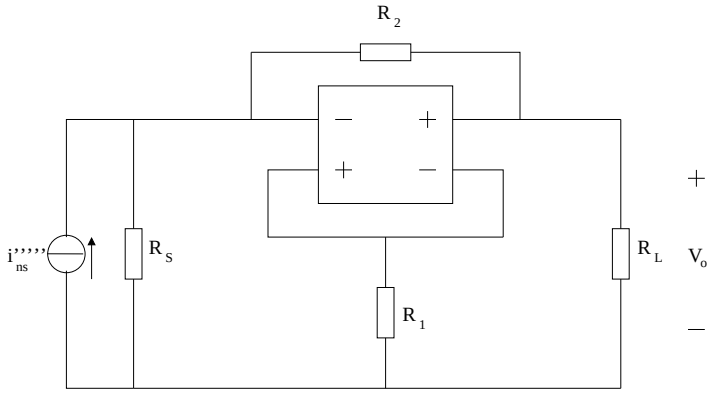

Figure A.50: Nullor based two-loop (B) amplifier. Current to Voltage. Step 10.

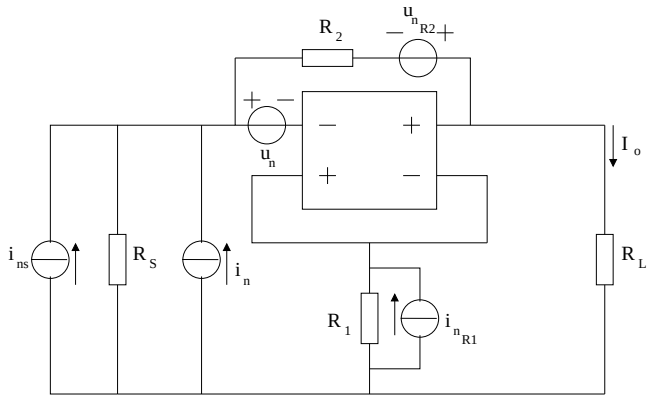

Figure A.51: Nullor based two-loop (B) amplifier. Current to Current.

Step 2. The noise current source i_{ns} and i_n are in parallel, these two sources are reduced into a single noise current source $i'_{ns} = i_{ns} + i_n$. Figure A.52.

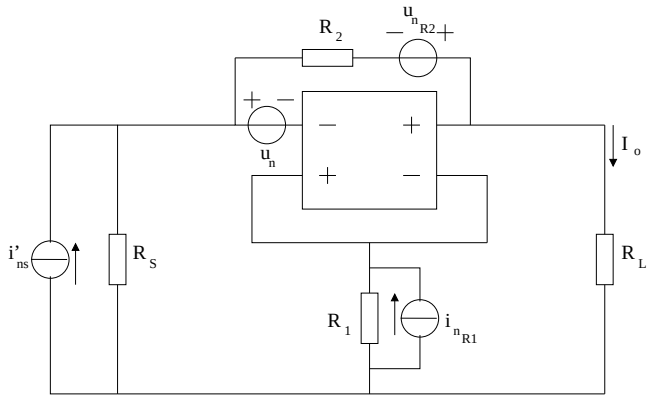

Figure A.52: Nullor based two-loop (B) amplifier. Current to Current. Step 2.

Step 3. V-shift is performed on u_n and is moved to the input branch of the amplifier and in series with $u_{n_{R_2}}$. Figure A.53.

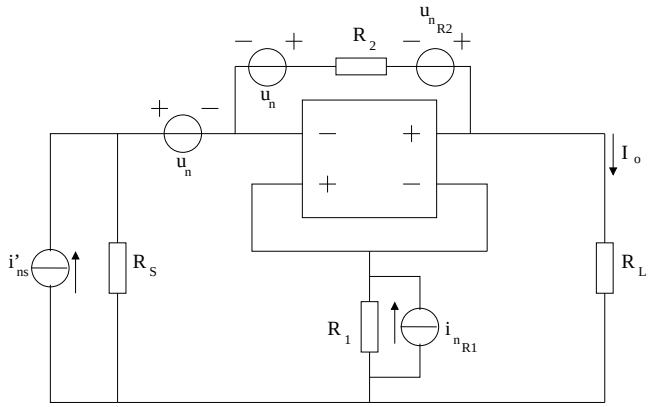

Figure A.53: Nullor based two-loop (B) amplifier. Current to Current. Step 3.

Step 4. For the source placed at the input branch the Norton-Thevenin transformation is applied in order to convert it into a noise current source $-u_n/R_S$. The other noise voltage source is reduced and becomes $u'_{n_{R_2}} = u_{n_{R_2}} + u_n$. Figure A.54.

Step 5. Since i'_{ns} is in parallel with $-u_n/R_S$ a reduction into a single source is performed giving as result $i''_{ns} = i'_{ns} - u_n/R_S$. For the source $u'_{n_{R_2}}$, Norton-Thevenin transformation takes place and this source is converted to $i'_{n_{R_2}} = u'_{n_{R_2}}/R_2$. Figure A.55.

Step 6. I-shift over $i'_{n_{R_2}}$ places one source at the input and one at the output ports of the amplifier. The one at the input is in parallel to i''_{ns}. Figure A.56.

Step 7. The noise current sources at the input are reduced into a single source to become $i'''_{ns} = i''_{ns} - i'_{n_{R_2}}$. Figure A.57.

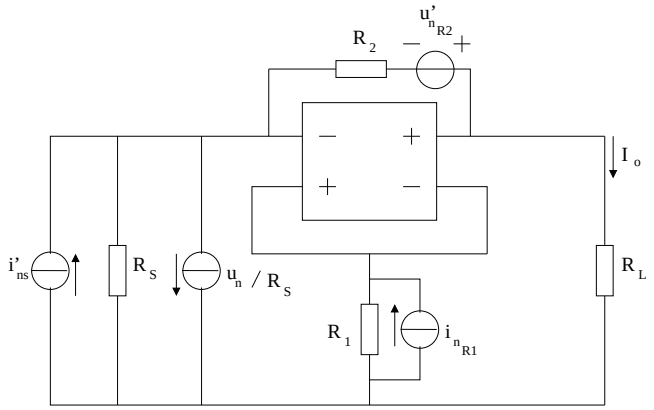

Figure A.54: Nullor based two-loop (B) amplifier. Current to Current. Step 4.

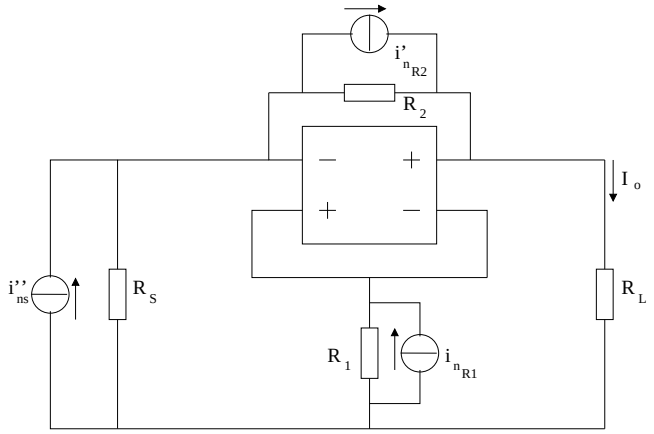

Figure A.55: Nullor based two-loop (B) amplifier. Current to Current. Step 5.

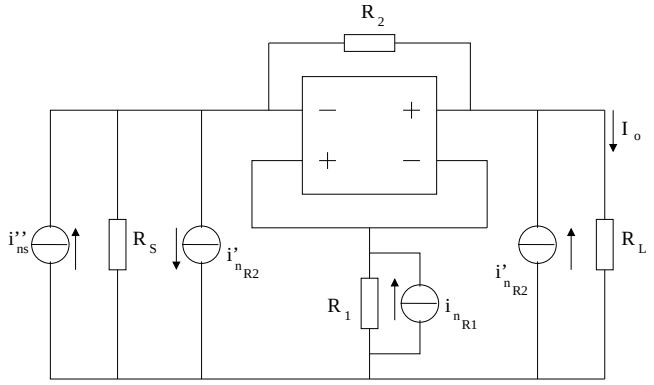

Figure A.56: Nullor based two-loop (B) amplifier. Current to Current. Step 6.

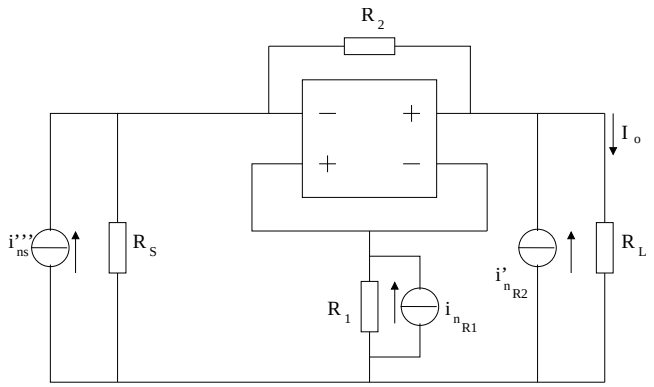

Figure A.57: Nullor based two-loop (B) amplifier. Current to Current. Step 7.

Step 8. For the source $i_{n_{R_1}}$ an I-shift is applied and places one source at the output port of the nullor and another at the output port in parallel to $i'_{n_{R_2}}$. The contribution of the noise current source at the output port of the nullor is zero, this is possible by properties of the nullor, therefore is discarded. Figure A.58.

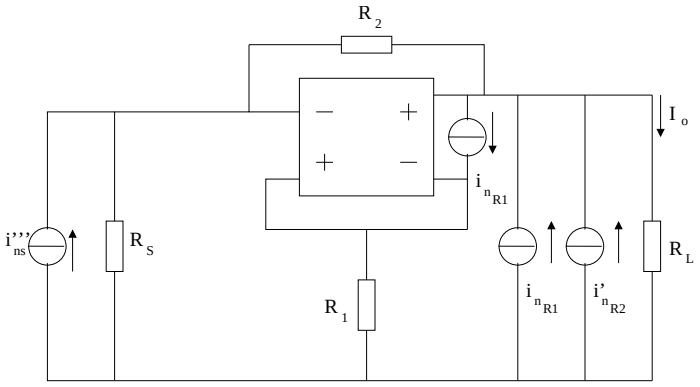

Figure A.58: Nullor based two-loop (B) amplifier. Current to Current. Step 8.

Step 9. At the output port the two noise current sources are reduced into a single source by $i''_{n_{R_2}} = i'_{n_{R_2}} + i_{n_{R_1}}$. Figure A.59.

Step 10. By Two-port transformation is possible to move $i''_{n_{R_2}}$ to the input port. This source becomes, by the transformation properties, in a voltage source $Bi''_{n_{R_2}}$ and a current source $Di''_{n_{R_2}}$. The latter is in parallel to i'''_{ns} and because the input source is current, the $Bi''_{n_{R_2}}$ source must be transformed. Figure A.60.

Step 11. $Di''_{n_{R_2}}$ source is added to i'''_{ns}, the result is given by $i''''_{ns} = i'''_{ns} + Di''_{n_{R_2}}$. Figure A.61.

Step 12. As stated before the $Bi''_{n_{R_2}}$ source must be converted into a current source. This convertion is performed by the Norton-Thevenin transformation. Now the value is $Bi''_{n_{R_2}}/R_S$. Figure A.62.

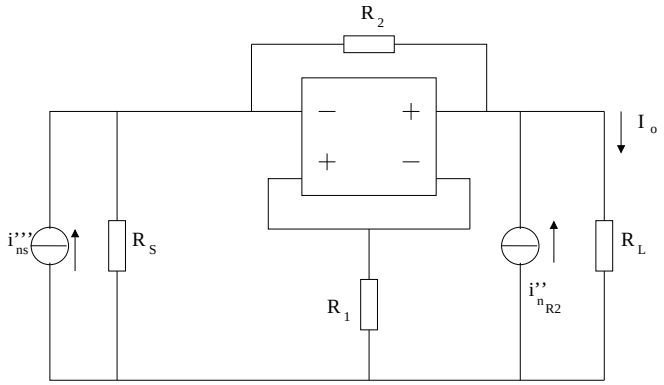

Figure A.59: Nullor based two-loop (B) amplifier. Current to Current. Step 9.

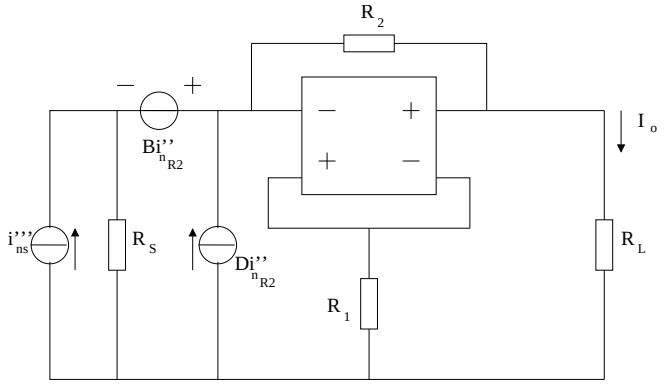

Figure A.60: Nullor based two-loop (B) amplifier. Current to Current. Step 10.

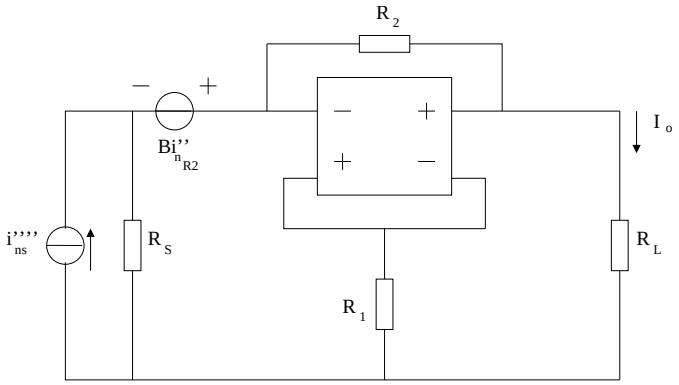

Figure A.61: Nullor based two-loop (B) amplifier. Current to Current. Step 11.

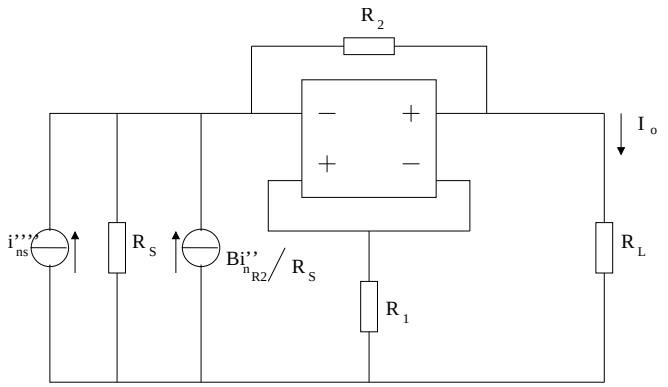

Figure A.62: Nullor based two-loop (B) amplifier. Current to Current. Step 12.

Step 13. The final step is to reduce $Bi''_{n_{R_2}}/R_S$ with i''''_{ns} and then all noise sources are placed outside the amplifier. Figure A.63.

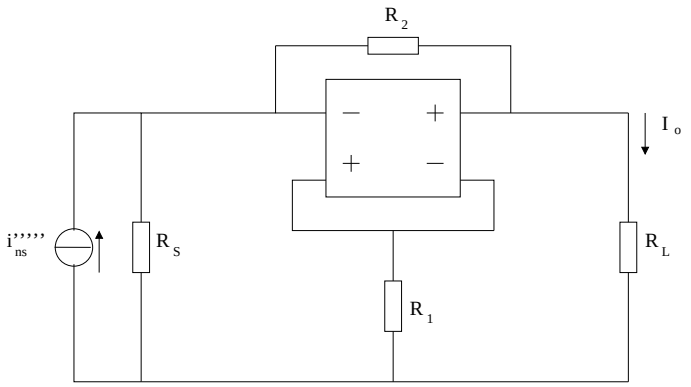

Figure A.63: Nullor based two-loop (B) amplifier. Current to Current. Step 13.

Appendix B

Chain Matrix Calculation

The Chain-matrix calculation is performed by means of applying certain conditions to some circuit and solving the MNA of the circuit, the values found represent the parameters ABCD of the circuit (this is why the Chain-matrix is also known as ABCD-matrix).

It is possible to automate this calculation in any programming language. Nevertheless, the complexity of the calculations makes it a tough task for programming languages like C or C++. The main reason is that the size of the matrix from wich it is based the calculation varies depending on the number of elements that makes up the circuit. Programming languages like C/C++ must dimension the array beforehand or create the array in a dynamic way, that is, by means of manipulating memory locations using pointers. For a novice electronic designer is quite difficult to create such kind of programs.

By means of the MAPLE [67] software it is possible to create functions capable to automate the MNA and Chain-matrix calculations. It has been developed a library for MAPLE called *kmna* that allows the calculation of the Chain-matrix and the impedance matrix for any circuit. The advantage of this library is the ability to analize a "sourceless" circuit, this means, a circuit is depicted by a netlist but it is not necessary to provide either a source and a load. The devices that this library can process are shown in Table B.1.

The operation of the library is quite simple. First it is necessary to indicate where the library is located, it could be placed somewhere in the user's directory or in a globall location if it was installed by the root user. This line indicates where is located the library:

```
libname:="<path>/lib",libname:
```

By this command the library is loaded into MAPLE and it will be ready to use:

```
with(kmna):
```

As for the input file, the file that contains the netlist, it is necessary to provide it using this format:

```
NetList:=[[name1,name2,...],
          [[position1A,position1B],[position2A,position2B],...],
          [value1,value2,..]]:
```

Keyword	**Device**
C	Capacitor
D	Diode
E	Voltage Controlled Voltage Source
F	Current Controlled Current Source
G	Voltage Controlled Current Source
H	Current Controlled Voltage Source
I	Independent Current Source
K	Coupled Inductors
L	Inductor
M	MOSFET Transistor (Basic Pi Model)
N	Nullator
O	Norator
Q	BJT Transistor (Basic Pi Model)
P	Nullor
R	Resistor
S	Short Circuit
T	Ideal Transformer
V	Independent Voltage Source
Y	Admittance
Z	Impedance

Table B.1: Devices supported by the kmna library for MAPLE.

Notice that the keyword *NetList* ought to be written in this way to indicate that a circuit description is provided, otherwise an error is produced and the process is aborted. The field **name***x* refers to the identifier given to the device. It must start with any of the given keywords provided in Table B.1 followed with one or more letters or numbers. The words **position***x***A** and **position***x***B** are the node numbers where the device is placed within the circuit topology. Finally, **value***x* is the numericall value given to the device. An example is given as follows:

```
NetList:=[[rs,cpi2,rpi2,cpi3,rpi3,gm3,RL,R1,R2],
          [[1,0],[1,0],[1,0],[5,3],[5,3],[5,3,2,3],[3,0],[2,0],[2,1]],
          [12500,1.424415E-11,2.069135E+04,2.936597E-11,8.621395E+03,
           1.159905E-02,1.000000E+04,56,2744]]:
```

Besides the netlist description it is necessary to provide the location where the input and output ports are located. The advantage to provide in a separate statement the location of the input and output ports is that it not only refers to the obvious input and output signal paths. For instance, the LP-product calculation ports can be given like the following:

```
Ports:=[[3,5],[1,0]]:
```

this refers that the input port is located between the node 3 and 5 while the output port, in consequence, is taken between the node 1 and 0. Keep in mind that node 0 is referred as the *grounding node* or the *reference node*.

To show the way this useful library works a simple example is provided. The calculation for a Chain-matrix basic nullor-based voltage amplifier (Figure B.1) is performed.

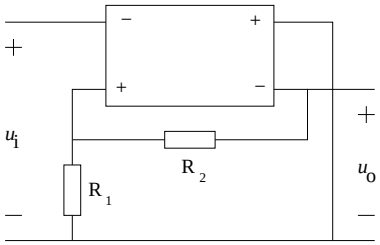

Figure B.1: Basic nullor-based voltage amplifier.

The netlist for this circuit is as follows:

```
NetList:=[[P1,R1,R2],[[1,2,0,4],[2,0],[4,2]],[N1,1,1]];
Ports:=[[1,0],[4,0]];
```

it is saved in a file named *nullor_a.cir*.

First step is to load MAPLE, clear all previous values and load the library:

```
restart:
libname:="/home/rsheissa/maple/kmna/lib",libname:
with(kmna):
```

Second step is the loading of the netlist, the library is programmed in such a way that allows the parsing of the netlist to create the MNA matrix. It also identifies where the "dummy" voltage, current and impedances are placed to obtain the Chain-matrix. The following shows the loading of the netlist into the variable (*test1)*, then the MNA matrix is also shown:

```
test1:=read_arch('/home/rsheissa/maple/kmna/nullor_a.cir'):
```

$$\left[\begin{bmatrix} Rv2 & 0 & 0 & -Rv2 & 0 & 0 & 0 & 0 \\ 0 & R2^{-1}+R1^{-1} & -R2^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & -R2^{-1} & Rv1+R2^{-1} & 0 & -Rv1 & -1 & 0 & 0 \\ -Rv2 & 0 & 0 & Rv2 & 0 & 0 & 1 & 0 \\ 0 & 0 & -Rv1 & 0 & Rv1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}\right.$$
$$\begin{bmatrix} I1 \\ 0 \\ I2 \\ 0 \\ 0 \\ 0 \\ V1 \\ V2 \end{bmatrix}$$
$$\left.\begin{matrix} 3 \\ 8 \end{matrix}\right]$$

Third step is to find the MNA matrix for the circuit, it is done issuing the code:

```
K1:=chain_matrix(test1);
```

the result is as follows:

$$\begin{bmatrix} \frac{R1}{R1+R2} & 0 \\ 0 & 0 \end{bmatrix}$$

It is also possible to obtain the impedance matrix for the circuit. As with the Chain-matrix, it is necessary to save the matrix in a variable (*Z1)* which produces:

```
Z1:=z_matrix(K1);
```

$$Z1 = \begin{bmatrix} \infty & \infty \\ \infty & \infty \end{bmatrix}$$

As it can be seen the impedance matrix for a nullor-based voltage amplifier is infinite. This result is coherent and proves the ideality of the nullor device.

This library can be employed within the graphical user inteface (GUI) of MAPLE or it can be run in the command line interface (CLI) provided by this tool. It is possible to write a "front end" in any programming language that calls the CLI interface of MAPLE to perform some calculations, save the results in a file and the program later retrieves the results to process them and show them in any way that the programmer wants.

In this thesis the latter approach was employed to find the output impedance of the amplifier and the open loop poles. Without this library it would have been harder to obtain these very important results.

Appendix C

Code Compilation and Upgrade

Two important issues to be resolved while developing a piece of software are the way it could be keep updated or the possibility to add new modules. In this appendix both concerns are discussed and the way to compile the code is also shown.

C.1 How-To Compile the Code

The following prerequisites needs to be satisfied in order to guarantee a succesful compilation of the code:

1. GNU Make version 3.81. This is a program that determines which pieces of a large program need to be recompiled and issues the commands to recompile them, when necessary.

2. Qt design software, it must be version 3.x. It is not guaranteed that it could compile using the version 4.x.

3. GNU C++ version 4.1. It is also possible to compile the tool using versions 3.3 or 3.4.

4. CLN library (*Class Library for Numbers*) version 1.1.13. This is a library written in C++ for computations with all kinds of numbers.

5. GiNaC symbolic framework. GiNaC (which stands for "GiNaC is Not a CAS (Computer Algebra System)") is a library for doing symbolic (i.e. non-numeric) computation directly in the C++ programming language. It is necessary to install the CLN library first.

6. Maple version 9 or 10. This software is not required to compile the program but is needed to perform various vital calculations.

The first step is to install the Qt software. It can be downloaded from the Qt homepage; for Solaris and Linux systems it is recomended to download the source package (*for instance qt-x11-free-3.3.7.tar.bz2*). It is out of the scope of this work to detail the installation process, to do so please refer to the documentation included in the package. To install the GiNaC

framework it is necessary to install the CLN library first. The installation instructions for both packages are included in their packages.

After the required packages are already installed, the next step is to create the *Makefile* file needed to compile all the required files of the tool. If there is already a *Makefile* in the development directory it is advised to erase it in order to guarantee a fresh environment. To create the file issue the following command:

```
qmake -o Makefile descad_wizard.pro
```

For a complete explanation on how to create a project file it is advised to refer the Qt documentation [71]. The previous command creates the *Makefile* from the descad_wizard project file. Once the file is created, to compile the whole project is performed by issuing the command:

```
make
```

It will take a while to compile the whole project, it depends on the type of processor and the available memory. After the compilation is concluded an executable program is created and the tool is ready to be used.

Warning: if for some reason the command

```
make clean
```

is issued, it is necessary to edit the file *bwcap.h* and add this code after the *qdialog* include statement

```
#include "transistor.h"
```

this is an issue found in the Qt package and not a flaw in the design and implementation of the tool.

C.2 Updating Code

The DESCAD wizard tool is comprised of several files each one serving a specific purpose. There is still the possibility that the modification of certain line or function could break the compilation of the tool. Each file is thoroughly explained, as much as possible, on the operation of each function.

It is possible to modify the code of any of the files with the exception of the *project* file. This can be done either by editing directly the file or overwriting it. As a cautionary measure it is recommended to create a backup of every file that is modified. It is strongly advised to place the adequate comments before the modified function or code segment.

Once the modification process has finished the code can be compiled by this command:

```
make
```

it is not necessary to update the project file since no new files were added.

C.2.1 Adding a New File

The new elements that can be added into the CAD tool can be either code files or graphical user interface (*GUI*) objects. GUI objects can be new windows within the project or new interfaces for new modules.

For the code files, insert them into the CAD tool project is very simple using the *Qt Designer* tool. A complete description about how this tool operates can be found in the Qt manual [49] and a thorough explanation on how to operate it is out of the scope of this work. The following explanation takes into account that the tool has been already loaded and the project opened (Figure C.1). Select the option *add a file to the project* from the file browser menu located at the right of the screen, select the file(s) to be added into the project (Figure C.2). It is possible to load multiple *cpp* files or *h* files at once. Once all the additional files are loaded into the project the next step is to save it in order to keep the changes. Now, it is necessary to regenerate the Makefile since new files need to be included in the final executable archive. To generate a new Makefile these steps need to be followed:

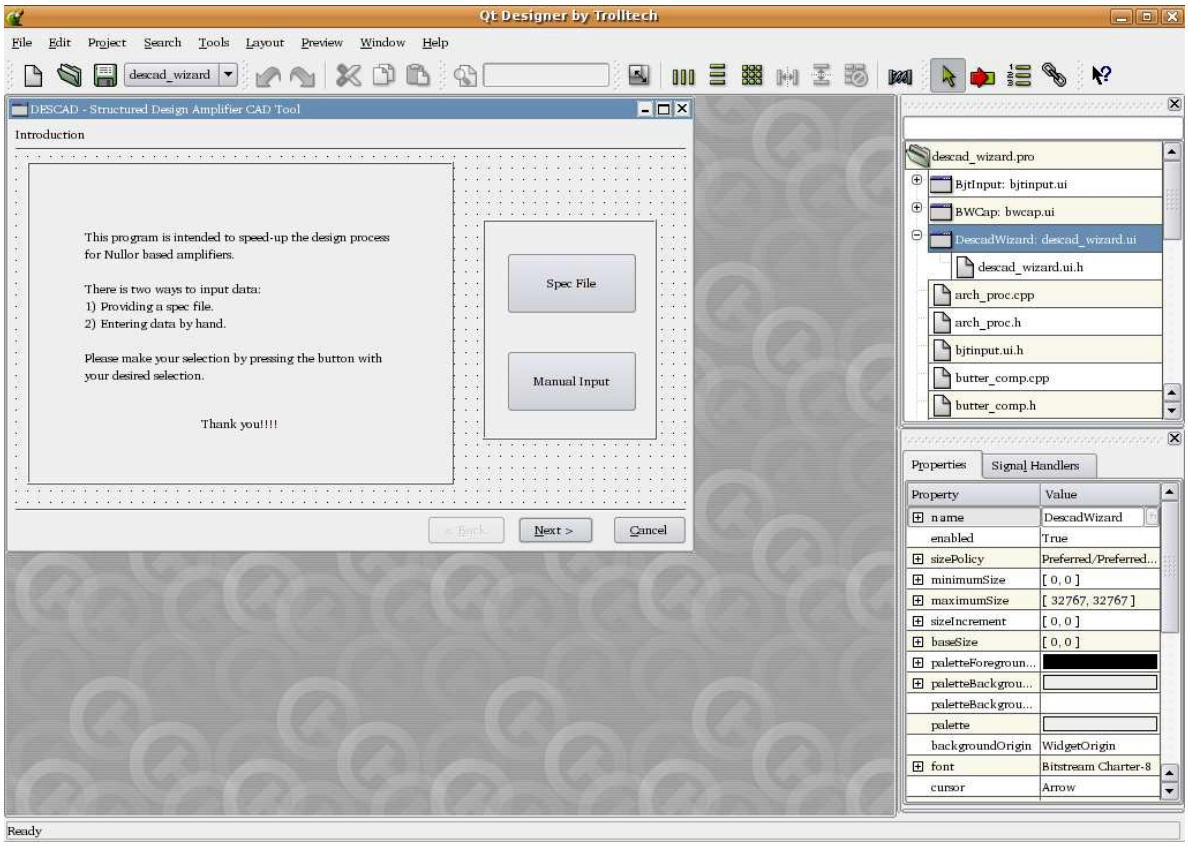

Figure C.1: Project main window.

1. Remove the old makefile issuing the command

   ```
   rm Makefile
   ```

2. The new makefile is created by typing

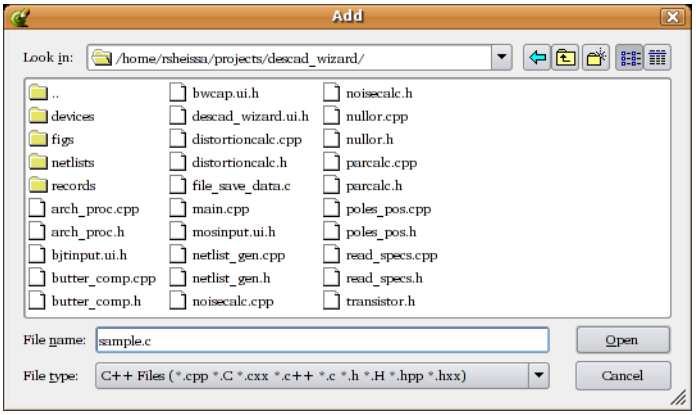

Figure C.2: Adding new file(s) to the project.

```
qmake descad_wizard -o Makefile
```

3. Rebuild the project using the command

```
make
```

The resulting executable file will include the changes or upgrades added to the project. Since it is a GUI tool most of the changes will not be visible unless some kind of feedback was programmed either by displaying it within a selected window or directly to the command line.

C.2.2 Adding a New Window

Adding a new window into the project is a task that can be done in several ways, but for the unexperienced designer the easiest way to perform this task is using the *Qt Designer* software [72]. The main window of the Qt Designer tool is shown in Figure C.1.

First step is to click with the right button of the mouse on the first page, which is the default behaviour, of the project and select the option *Add Page*. This new page will become the last window of the project (Figure C.3).

This new window can be populated with all kind of graphical aids like dropdown menus, text edit boxes, buttons and more. Please keep in mind that a bloated window may cause confusion to the user, the best method is to keep the interface as simple as possible. Once the "drawing phase" has concluded, the next step is to write the code that will provide the desired functionality to the graphical objects placed in the page. The functionality will be provided by a function which can be created by clicking **Edit→Slots** on the upper toolbar (Figure C.4). Press the button *New Function* to create the function, in the *Function* textbox names the function and in the *Return type* textbox provides the name of the returning value. It is advisable to leave the other settings untouched. To save the new function press the button *OK*.

The last step is to link the graphical element to a certain function. Clicking the upper toolbar in **Edit→Connections** will show all the already functions and their related graphical

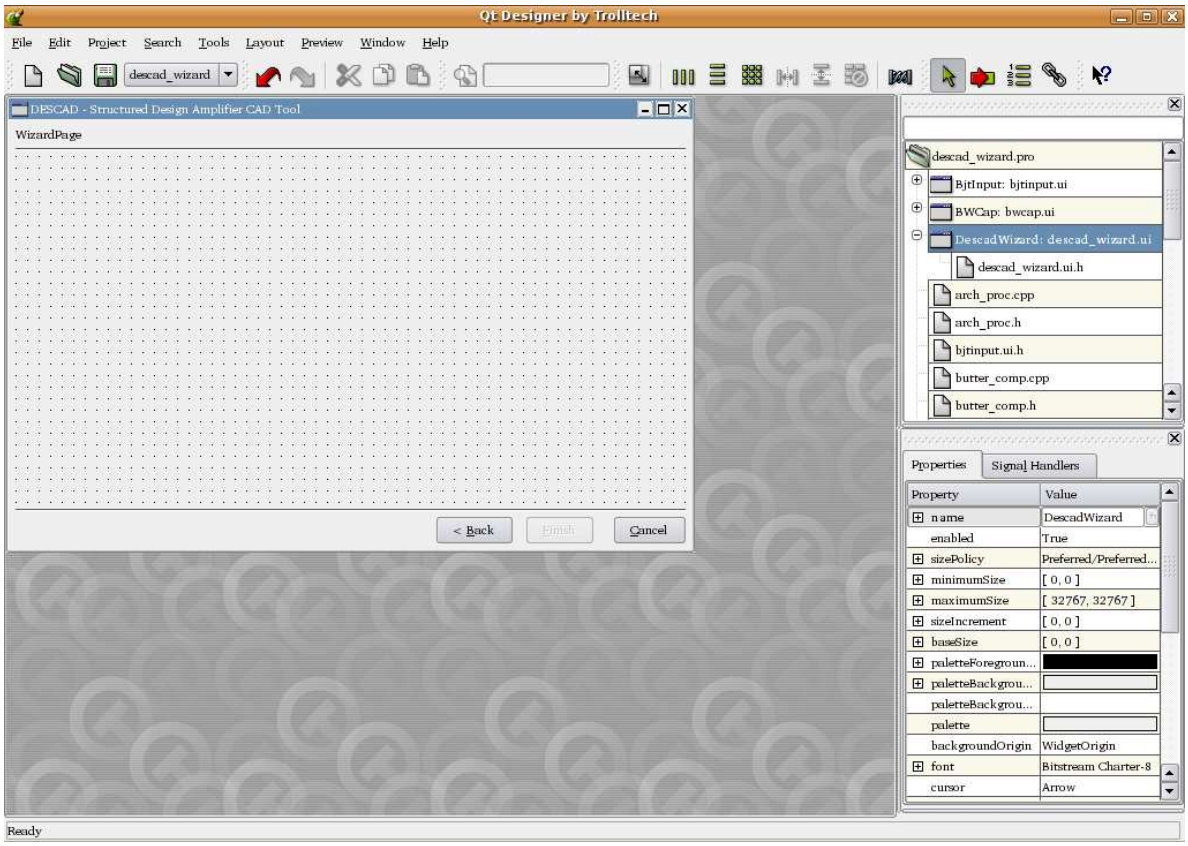

Figure C.3: A new window is added to the project.

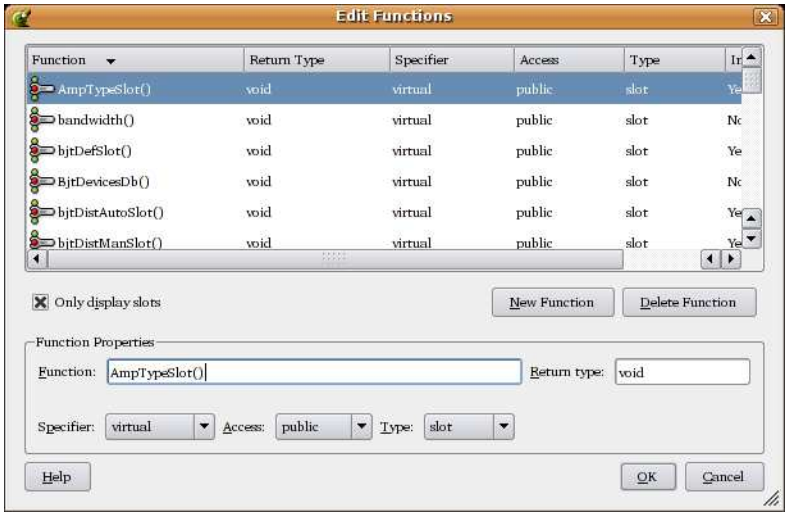

Figure C.4: Function editor window.

element (Figure C.5). Pressing the *New* button will create a new instance at the bottom of the list; the first option is to select the *Sender*, that is, an specific graphical element will do something giving a certain condition. Clicking on this field will provide a dropdown menu showing all the graphical elements of the project, be careful to avoid duplicities because the last selection will be the one to be kept. The next field refers to the kind of action provided by the element, for combo boxes, dropdown menus and buttons the most common option is to selection the action *clicked()*. For other kind of elements there are other available options to choose from, it will depend on the selected element. The third field is the *Receiver* option, this field refers to which class will receive the generated event. The *Receiver* field will usually be the main class which in this example is *DescadWizard*. Finally, the fourth field will show the name of the function that will process the action generated by the *Sender*. For a detailed explanation on each field of the *View and Edit Connections* window please refer to [49].

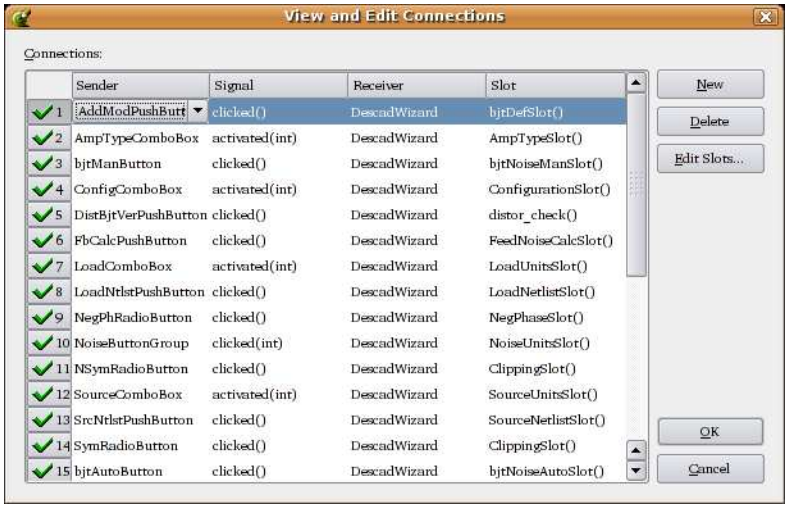

Figure C.5: Connections editor window.

At this stage the new window has all the desired functionallity, nevertheless it is required to place it in a new position within the wizard progression. This goal can be achieved by selecting the desired position in the field *currentPage* within the Properties tab shown in Figure C.6. First to be done is to modify the page number for the page that is going to be shifted because the designer tool will show an error message if two pages have the same number. Then, the new window is set with the desired number and it will be placed automatically in the right position. To save all the changes click on the upper toolbar on **File→Save All** option. To compile all the changes follow the guidelines provided in C.2.

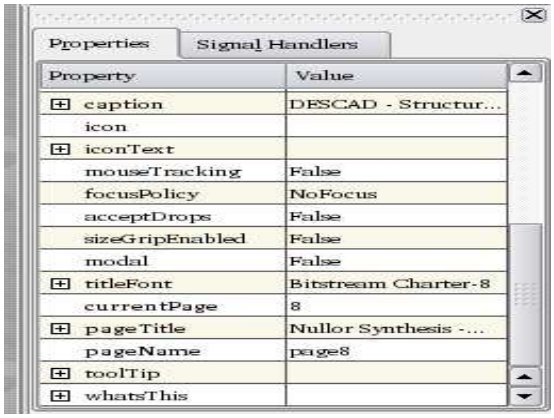

Figure C.6: Properties tab to select a new position within the wizard.

Bibliography

[1] C. Verhoeven, A. Staveren, G. Monna, and M. Kouwenhoven, *Strutured Electronic Design, Negative-feedback Amplifiers*. Delftese Uitgevers Maatschappij, 2001.

[2] T. Holden, *Knowledge Based CAD and Micro-electronics*. Elsevier Science Publishers, 1987.

[3] T. J. Barnes, *Electronic CAD Frameworks*. Kluwer Academic Publishers, 1992.

[4] T. Barnes, "Codem: The cooperative development model," 1988, unpublished paper, National Semiconductor Corporation.

[5] A. Doboli, A. Nunez, N. Dhanwada, S. Ganesan, and R. Vemur, "Behavioral synthesis of analog systems using two-layered design space exploration," in *Design Automation Conference*, 1999, pp. 951–957.

[6] N. Dhanwada, A. Nunez, and R. Vemuri, "Hierachical constraint transformation using directed interval search for analog systems," in *DATE '99*, 1999, pp. 328–335.

[7] R. Harjani, R. Rutenbar, and L. Carley, "Oasys: A framework for analog circuit synthesis," *IEEE Trans. on Computer-Aided Design*, vol. 8, no. 12, pp. 1247–1266, December 1989.

[8] J. D. Lohn and S. P. Colombano. (1998) Automated analog circuit synthesis using a linear representation.

[9] N. Dhanwada, A. Nunez-Aldana, and R. Vemuri, "Automatic constraint transformation with integrated parameter space exploration in analog system synthesis," in *ASP-DAC*, 1999.

[10] T. Quarles, A. Newton, D. Perderson, and A. Sangiovanni, *SPICE3 Version 3f3 User's Manual*, University of California, Berkeley, Ca., 94720, May 1993.

[11] P. Nenzi *et al.*, *NGSPICE User Manual*, IEEE, Roma, Italy, March 2001.

[12] M. Herniter, *Schematic Capture with Cadence PSpice*, 2nd ed. Prentice-Hall, 2002.

[13] M. Rashid, *Introduction to PSpice Using OrCAD for Circuits and Electronics*, 3rd ed. Prentice-Hall, 2003.

[14] *Star-Hspice Manual*, Avant! Software, 2001, Release 2001.2.

[15] A. Davis, *The GNU Circuit Analysis Package User Manual*, 2004, version 0.34.

[16] *APLAC Circuit Simulation & Design Tool 7.80 Manual*, APLAC Solutions Corporation, Helsinki, Finland, 2002.

[17] *CADENCE Reference Manual*, Cadence Design Systems, 2003.

[18] A. Hossain, *Computer Aided Electronic Circuit Board Design and Fabrication: Using OrCAD/SDT and OrCAD/PCB Software Tools*, 1st ed. Prentice-Hall, 1995.

[19] *gEDA: GNU Electronic Design Assistant*, Univeristy of Seoul, Seoul, Korea, 2005.

[20] A. Heck, *Introduction to Maple*. New York: Springer-Verlag, 1993.

[21] M. L. Abell and J. P. Braselton, *The Mathematica Handbook*. AP Professional, 1992.

[22] J. O. Attia, *PSPICE and Matlab for Electronics*. CRC Press Inc., 2002.

[23] J. Broz *et al.*, *Analog Insydes Manual*, Fraunhofer ITWM, Germany, 2005, version 2.1.

[24] J. Riel, *syrup: A Symbolic Circuit Analyzer for Maple*, August 2002, version 7.1-01.

[25] L. O. Roque, "Lenguaje de alto nivel para la descripción y simulación de circuitos analógicos," Ph.D. dissertation, INAOE, 2003.

[26] J. Koza, F. B. III, D. Andre, and M. Keane, "Automated design of both the topology and sizing of analog electrical circuits using genetic programming," in *Artificial Intelligence in Design*, J. Gero and F. Sudweeks, Eds. Dordrecht: Kluwer Academic, 1996, pp. 151–170.

[27] N. Azizi. (2000) Automated analog circuit design using genetic algorithms.

[28] J. Shao and R. Harjani, "Macromodeling of analog circuits for hierarchical circuit design," in *IEEE ICCAD*, November 1994, pp. 656–663.

[29] H. Chang, A. Sangiovanni-Vincentelli, F. Balarin, E. Charbon, G. Jusuf, E. Liu, E. Malavasi, R. Beff, and P. Gray, "A top-down, constraint-driven design methodology for analog integrated circuits," in *Custom Integrated Circuits Conference*. IEEE, May 1992, pp. 841–846.

[30] J. Shao and R. Harjani, "Feasibility region modelling of analog circuits for hierarchical circuit design," in *Midwest Symposium on Circuits and Systems*. IEEE, 1994.

[31] C. Verhoven, J. Westra, and A. van Staveren, *Analog Circuit Design*, J. Huising and et al, Eds. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1996.

[32] E. Norholt, "Strategies and routines in analog design," in *Proceedings of the Workshop Advances in Analog Circuit*, April 1992, pp. 369–396.

[33] J. Stoffels, "Automation in high-performance negative feedback amplifier design," Ph.D. dissertation, Delft University of Technology, 1988.

[34] E. Nordholt, *Design of High Performance Negative-feedback amplifiers*. Elsevier Scientific, 1983.

[35] J. Davidse, *Analog Electronic Circuit Design*. Prentice Hall, 1991, new York.

[36] J. Stoffels and C. Reeuwijk, "Ampes: A program for the synthesis of high-performance amplifiers," in *Proceedings of the European Conference on Design Automation*, March 1992, pp. 474–479.

[37] A. van Staveren, C. Verhoeven, and A. van Roermund, "The influence of the reverse early effect on the performance of bandgap references," *IEEE Transactions on Circuits and System I*, vol. 43, no. 5, pp. 418–421, May 1996.

[38] ——, "The design of low-noise bandgap references," *IEEE Transactions on Circuits and System I*, vol. 43, no. 4, pp. 290–300, April 1996.

[39] C. Verhoeven, "First-order oscillators," Ph.D. dissertation, Delft University of Technology, The Netherlands, 1990.

[40] C. Boon, "Design of high-performance negative-feedback oscillators," Ph.D. dissertation, Delft University of Technology, The Netherlands, 1989.

[41] J. Westra, R. Godjin, C. Verhoeven, and A. van Roermund, "Coupled relaxation oscillators with highly stable and accurate quadrature outputs," in *Proceedings of the 1st IEEE-CAS Region 8 Workshop on Analog and Mixed IC Design*, Pavia, Italy, September 1996, pp. 32–35.

[42] J. Voorman, "The gyrator as a monolithic circuit in electronic systems," Ph.D. dissertation, University of Nijmegen, Nijmegen, The Netherlands, 1977.

[43] B. Nauta, "Analog cmos filter for very-high frequencies," Ph.D. dissertation, Delft University of Technology, Enschede, The Netherlands, 1991.

[44] G. Groenewold, "Optimal dynamic range integrated continous-time filters," Ph.D. dissertation, Delft University of Technology, The Netherlands, 1992.

[45] G. Monna, "Design of low-voltage integrated filter-mixer systems," Ph.D. dissertation, Delft University of Technology, Delft, The Netherlands, 1996.

[46] C. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–432 and 623–656, July 1948.

[47] D. Smith *et al.*, "Designing the star user interface," Byte, April 1982, pp. 242-282.

[48] M. Vance, *Glade Manual V2.0*, GNOME Documentation Project, California, USA, 2004.

[49] *Qt Reference Documentation*, Trolltech, Oslo, Norway, 2005, ver. 3.3.4.

[50] D. G. Peterson, "Noise performance of transistors," *IRE Trans. Electron Devices*, vol. ED-4, pp. 296–303, May 1962.

[51] A. Van der Ziel, *Fluctuation Phenomena in Semiconductors*, 1959, London:Butterworth.

[52] A. van der Ziel, "Thermal noise in field-effect transistors," *Proc. IRE*, vol. 50, pp. 1808–1812, August 1962.

[53] A. G. Jordan and N. A. Jordan, "Theory of noise in metal-oxide semiconductor devices," *IEEE Trans. Electron Devices*, vol. ED-12, pp. 148–156, March 1965.

[54] H. W. Ott, *Noise Reduction Techniques in Electron Systems.* John Wiley & Sons, 1976.

[55] Y. Netzer, "The design of low-noise amplifiers," *Proc. IEEE*, vol. 69, no. 6, pp. 728–742, June 1981.

[56] V. Radeka, "1/f noise in physical measurements," *IEEE Trans. Nuclear Sci.*, vol. NS-16, pp. 17–35, October 1969.

[57] R. A. Dukelow, "An experimental investigation of very low frequency semiconductors noise," Ph.D. dissertation, California Inst. Technol., Pasadena, 1974.

[58] G. Palumbo and S. Pennisi, *Feeback Amplifiers.* Kluwer Academic Publishers, 2002.

[59] W. Sansen, "Distortion in elementary transistor circuits," *IEEE Trans. on Circuits and Systems-part II*, vol. 46, no. 3, pp. 315–324, March 1999.

[60] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, 4th ed. Oxford University Press, 1998.

[61] W. H. Hayt Jr. and J. Kemmerly, *Engineering Circuit Analysis.* McGraw-Hill Inc., 1992.

[62] R. L. Boylestad and L. Nashelsky, *Electronic Devices and Circuit Theory*, 6th ed. Prentice-Hall Hispanoamericana, 1997.

[63] H. W. Black, 1937, U.S. Patent 2, 102, 761.

[64] L. P. Huelsman and P. E. Allen, *Introduction to the Theory and Design of Active Filters.* McGraw-Hill, 1980.

[65] G. Booch, *Object Oriented Design With Applications.* The Benjamin/Cummings Publishing Company, Inc., 1991.

[66] L. Joyanes-Aguilar, *Borland C++.* McGraw-Hill, 1993.

[67] B. W. Char, K. O.Geddes, and G. H. Gonnet, "Maple V language reference manual," Springer-Verlag, New York, 1994. ISBN/ISSN:0-387-97622-1.

[68] F. Svelto, E. Sacchi, F. Gatta, D. Manstretta, and R. Castello, "Cmos low-noise amplifier design," in *Low-power Design Techniques and CAD Tools for Analog and RF Integrated Circuits.* Kluwer Academic Publishers, 2001, pp. 251–265.

[69] T. Saari and M. Ritamäki, "446mhz low noise amplifier for a pmr446 receiver," Nokia, Tech. Rep., 2004.

[70] M. E. Miranda-Varela, "Formulación y solución de problemas de programación no lineal en el diseño estructurado de amplificadores," Master's thesis, Laboratorio Nacional de Informática Avanzada, Centro de enseñanza LANIA, 2006.

[71] M. Delheimer, *Programming with Qt.* O'Reilly Media, 2002.

[72] J. Blanchette and M. Summerfield, *C++ GUI Programming with Qt 3.* Prentice Hall, 2004.