New MOSFET Modeling Algorithms and Their Use in CAD of Analog IC Building Blocks

Kaustubha Mendhurwar

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montreal, Quebec, Canada

April 2008

Your file Votre référence
ISBN: 978-0-494-40891-9
Our file Notre référence
ISBN: 978-0-494-40891-9

ABSTRACT

New MOSFET Modeling Algorithms and Their Use in CAD of Analog IC Building Blocks

Kaustubha Mendhurwar

Analog integrated circuit (IC) design has undergone several technical advancements following Moore's law, and tends to become extremely challenging with the continued downscaling of the devices and supply voltages. However, not many sophisticated and detailed design tools are available to aid analog designers in exploiting the complete potential of these technical advancements. Most of the available commercial and in-house design tools model the basic building blocks (*e.g.* transistor) and employ these device models to predict the performance of a complete circuit/system. As such, accuracies of these device models are crucial in order to develop efficient design tools. Typically, accurate models could be complex, while simple models could be inaccurate. As such, new modeling algorithms leading to simple yet accurate device models and satisfactory design tools continued to be in great demand.

In this thesis, neural networks that offer advantages like simple calculations and a wide spectrum of applications, are employed for the modeling purpose. Firstly, new

modeling algorithms based on binning concepts that offer accurate device modeling over wider input parameter space of the problems that have outputs highly non-linear to one of the inputs are proposed. Both single and multi- dimensional modeling algorithms are developed, and illustrated through device modeling examples. A new neural modeling approach based on a correction model is then introduced for the first time to develop accurate device level models for highly non-linear input-output behaviours that are difficult/impossible to model with simple structures. The proposed approach simplifies the modeling process for a novice/inexperienced designer as it eliminates the need of in depth understanding of the neural network concepts by virtue of using well known simple 3-layer MLP networks. MOSFET modeling example confirms that the approach leads to accurate neural models while keeping the model structure simple.

Finally, device models developed using the aforementioned modeling algorithms are employed to build an accurate and extendable computer aided design (CAD) tool for the design of analog IC building blocks (*e.g.* current sources/sinks, single stage amplifiers, simple and cascode current mirrors, voltage divider, differential amplifier, and three stage operational amplifier).

Acknowledgements

I would like to express my sincere gratitude towards my supervisors Dr. Rabin Raut and Dr. Vijay Devabhaktuni for their tireless efforts in the advancement of my education, career and life goals. Their continued guidance and encouragement helped me overcome the usual teething troubles involved in the research. They provided me their valuable time and extensive technical knowledge from pillar to post of my thesis work.

I would also like to appreciate valuable suggestions and efforts of my colleagues in the CAD group at Concordia University, without whom, this work would not have been possible. I would like to make a mention for Rajasekhar Kakumani, Niladri Roy, Li Zhu, Arash Kashi, Navid Arbabi, Farzin Manouchehri, Mani Najamabadi, Ahmad Zbeeb, and Joshua Frankel for their continued support and friendship.

Last, but not the least, I would like to extend my heartfelt appreciation to my family (especially my brother Jay) and friends for their ongoing support and understanding.

To my loving parents

Rekha & Ashok

Table of Contents

List of Figures

List of Tables

List of Symbols and Abbreviations

η	Learning rate
kT/q	Thermal voltage
γ	Body effect constant
λ	Body effect parameter
$2\varphi_F$	Surface potential parameter
μ_n	Charge-carrier mobility
A/D	Analog to digital
AM	Amplitude modulation
ANN	Artificial Neural Network
A_V	Open loop gain
BSIM	Berkely short-channel IGFET model
BP	Binning parameter
CAD	Computer aided design
CMOS	Complementary metal oxide semiconductor
CNT	Carbon nano tube
Cox	Gate capacitance across the oxide per unit area
CPU	Central processing unit

D/A	Digital to analog
DA	Design automation
DC	Direct Current
E_{avg}	Aggregate error measure
E_C	Critical field
E_{user}	User-specified error
E_{worst}	Worst-case error
g_{ds}	Output conductance
g_m	Transconductance
g_{mb}	Body effect transconductance
GUI	Graphic user interface
I_{bias}	Biasing current
IC	Integrated circuits
I_{DS}	Drain source current
KBNN	Knowledge based neural networks
L	Effective gate length of the transistor
MLP	Multi layer perceptrons
MNT	Micro nano technology
MOS	Metal oxide semiconductor
MOSFET	Metal oxide semiconductor field effect transistor
NMOS	N channel metal oxide semiconductor
OPAMP	Operational amplifier

OTA	Operational transconductance amplifier
PMOS	P channel metal oxide semiconductor
V	Volts
V_{BS}	Bulk source voltage
V_{DD}	Supply voltage
V_{DS}	Drain source voltage
V_{Dsat}	Drain saturation voltage
V_{GS}	Gate source voltage
V_T	Threshold voltage
V_{T0}	Zero bias threshold voltage
W	Gate width of transistor
μA	Micro ampere
μm	Micro meter
μW	Micro watt

CHAPTER 1

INTRODUCTION

1.1 METAL-OXIDE SEMICONDUCTOR FIELD-EFFECT TRANSISTORS (MOSFETS) AND THEIR MODELS

Recent advances in MOSFET technology, such as the continued downscaling of the physical dimensions, use of higher electric fields and continuing decrease in the power supply voltage, have made their behavior highly useful yet complex. Unfortunately, these technological advancements have not been followed up with concurrent improvements in analog design approaches. Most of the analog design approaches are still employing the

methods based on much simpler first hand approximations of MOSFET behavior. This has created an unwanted scenario, in which the circuit designers are applying outdated methodologies on newer technologies. As such, designers may not be able to realize the entire potential of modern deep submicron complementary metal-oxide semiconductor (CMOS) technology [1].

Transistors are key components in modern circuit design. Transistor based circuits are used globally, not only in analog but in digital circuits as well. Furthermore, transistors are more often than not the basic building blocks on which performance of the entire circuit depends. Playing such an important role in a multitude of circuits, one would think that the perfect understanding of its operation would be paramount. Unfortunately, this is not the case. Several slow and cumbersome methods (to be discussed in detail in chapter 2) are available that are based on approximations/assumptions. These assumptions may not hold true in every situation but often assumed to be true. As such, the development of new modeling methods for this key component becomes important. Geometrical/physical parameters involved in transistor modeling are illustrated by the physical structure of an NMOS transistor, depicted in Fig. 1.1.

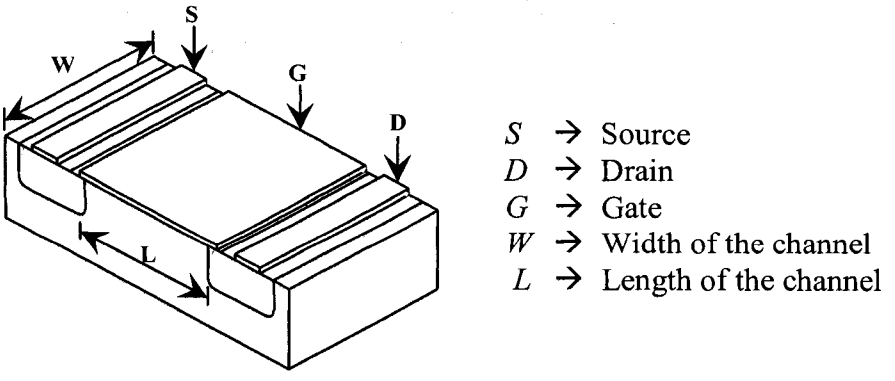


Figure 1.1 Physical structure of an NMOS transistor.

A typical design problem for analog designers is to determine functional relationship f such that

$$[L, W/L] = f(V_{BS}, V_{GS}, V_{DS}, I_D), \tag{1.1}$$

where, L and W represent length and width of the MOS transistor. V_{BS}, V_{GS}, and V_{DS} represent the bulk voltage, gate voltage, and drain voltage respectively, and I_D represent the drain current.

As a side note, it is imperative to acknowledge that the design of the MOSFET has the potential to affect the overall design of several widely used building blocks *e.g.* current sinks/sources, current mirrors, differential amplifier, *etc.* It is therefore substantial that the MOSFET be designed with utmost diligence. Device modeling is critical/vital to enabling design automation of circuits and systems.

1.2 Design Automation in Analog Domain

Analog design is known to be a knowledge-intensive, multiphase, and iterative task. It usually stretches over a significant period of time and is performed by designers with a large portfolio of skills [2]. Text books as well as publications may not be readily useful in exploiting good design techniques for successful analog circuit generation as these techniques reside mainly in the experience and expertise treasured by relatively very few analog designers. The advent of computers has led to what are known as Computer Aided Design (CAD) tools or design automation (DA) tools. Typically, a CAD or DA tool is a computerized program/software that assists circuit designers in the accomplishment of a design objective. The CAD tools have a property to automate a part/whole of the design

process. Silicon compilers fall in this category of CAD tools as they produce integrated circuit layouts straight from certain higher level specifications [3][4].

In the recent years, smaller feature sizes and higher scales of integration have resulted in an increased circuit design complexity. In order to deal with the design complexity, the need for automated design tools arises, as does the need for optimization tools to be able to automate several aspects of the design process while adhering to the tight process technology constraints [4]. Irrespective of the technological advancements in the analog domain, analog CAD tools still are in the nascent stages. In particular, in terms of design time, analog CAD tools lag considerably in comparison with the thoroughly detailed and highly sophisticated digital CAD tools. An example typically quoted is that while 90% of an integrated circuit may be digital with only 10% analog, most of the design time and effort is still devoted to the analog part.

Since the real-world signals are analog in nature, implementation of both analog and digital functionalities on the same chip has always been a necessity as well as a design challenge. Hence, for the efficient design of analog integrated circuits, present and future trend is to develop more robust industrial analog CAD tools. These tools are designed with primary focus on the evolution in areas like, circuit and system synthesis, symbolic analysis, automated layout generation, and testing and optimization of the circuit designs to meet critical specifications of the high-performance designs.

The analog integrated circuit (IC) design process is comprised of three major phases, namely (i) synthesis, (ii) design, and (iii) implementation. These three phases of analog integrated circuit design process are depicted, in detail, in Fig. 1.2.

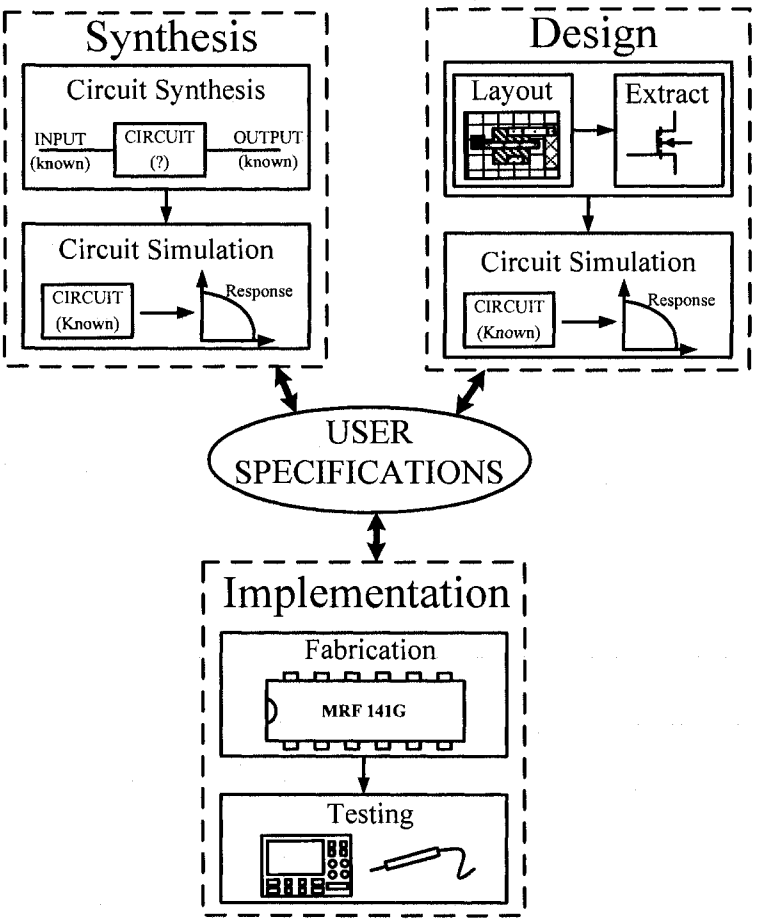


Figure 1.2 Analog integrated circuit design process.

(i) Synthesis: This phase focuses on identifying the circuit topology that satisfies given input-output behavior. This phase can be tiresome as sometimes there can be no solution and more often the solution is not unique. Analog circuits, especially the most useful and frequently needed, are rarely novel in the strict sense of the word [2]. Most often same building blocks are adjusted and tailored to suit the specific application goals. As such, the synthesis phase is not that crucial in analog IC design. Designers can deal with and can predict performance of circuits, comprising of fewer transistors, confidently owing to their design expertise/experience. Designers often make appropriate assumptions/guesses in terms of the initial circuit, based on their design experience and expertise.

(ii)Design: This phase deals with the actual design of the circuit obtained in the synthesis phase. In this phase, circuit designers have to find meaningful values for components of the circuit obtained from synthesis that satisfy the design goals. Simulators *e.g. HSPICE* contribute a great deal in this phase, as simulation is the only means to foresee the circuit response without actually fabricating the circuit. There are numerous design approaches (discussed in detail in chapter 2), and an appropriate one is selected depending on the design specifications. The design process is tedious as identifying the design variables that should be tuned, itself requires considerable design expertise. Furthermore, the decision about direction and amount of tuning relies on the designer's knowledge and experience on the circuit under consideration. As a result, designers invest enormous amount of time in fine tuning the design variables, in order to satisfactorily meet the user specifications. In conclusion, it may be noted that the design/optimization loop requires knowledge of a multitude of disciplines, and can be unending for a novice designer.

(iii) Implementation: This is the concluding phase of the process, where the designed circuit is fabricated and tested extensively for the given user specifications. Depending on the testing results, fabricated circuits are either sent for mass production or back to the designers for further fine tuning.

In essence, all of the above phases would benefit tremendously from research and development of CAD tools.

1.3 Objectives and Motivation

As mentioned earlier, this thesis is primarily motivated by an aspiration to simplify the complex and time consuming design automation process for analog circuits. Scarce efforts

involved in analog domain to augment the present status of available design automation tools, only aggravate the problem for analog designers. Research in the design automation area is inadequate to match the technological advancements and as a result, prevents the complete exploitation of advanced technologies. Dearth of design expertise in the analog domain is the major driving force for the need of design automation tools. Therefore, the principal objective of this thesis is to introduce some new modeling algorithms at the device level, to produce accurate device models, and employ those models at the circuit level to aid and enhance the design process. From an industry perspective, this work is practical as it intends to make the design process simple and technology independent for novice users. Detailed objectives of the thesis work are depicted in Fig. 1.3.

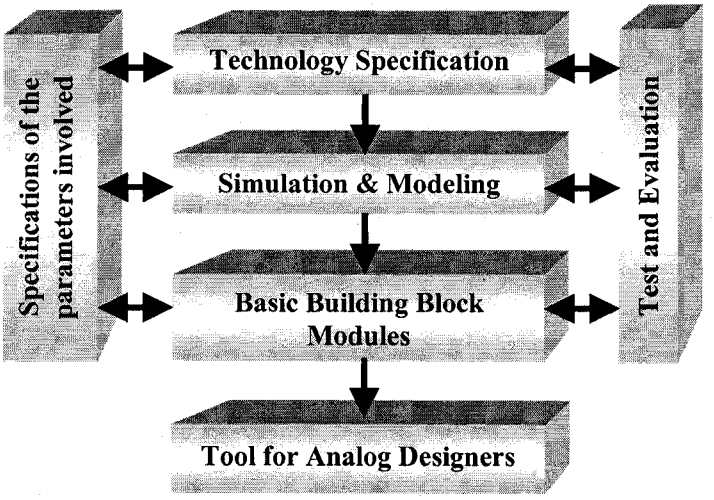


Figure 1.3 Flowchart showcasing the outline of the thesis.

1.4 THESIS OUTLINE

In essence, this thesis provides new modeling algorithms that help develop accurate and advanced device models. The device models are employed as basic building block modules at the circuit level leading to a CAD tool for analog designers to design basic to

complex circuits. Finally, application of this tool to 0.5 μm and 0.18 μm technologies is illustrated through several design examples.

Chapter 2 commences with some analog design history and importance of MOSFET modeling, followed by some background theory on commonly used MOSFET modeling approaches and artificial neural networks (ANN) that is relevant to the thesis objectives. In-practice modeling approaches are briefly described and the need for new modeling methods/techniques is discussed. Neural networks, as the potential modeling approach, is manifested through basic theory, their implementation areas, and the benefits they offer.

Chapter 3 introduces two novel MOSFET modeling algorithms based on the concept of binning. Single and multi-dimensional binning algorithms are proposed and showcased using neural networks as a case study. Proposed algorithms help model devices that show a relatively linear behaviour along certain axes and more non-linear behaviour along other axes. The chapter concludes with application of proposed algorithms to modeling problem on hand (*i.e.* MOSFET modeling).

Chapter 4 introduces a new neural network modeling approach based on a correction model concept for accurate modeling of devices/components. A detailed flow-chart of the proposed modeling approach is presented along with a pictorial depiction of the concept. The proposed approach helps model the problems that are difficult/impossible to model using the standard neural modeling approach. It has the potential to simplify the modeling process for a common user, without much background knowledge. In the final section of this chapter, the proposed approach is employed to design problem of a MOSFET.

Chapter 5 showcases the design tool, developed as a part of this thesis, employing the modeling algorithms presented in chapters 3 and 4, in detail with all its development

phases described briefly. Several design examples for 0.5 μm technology are provided along with the simulation verification results. In addition, the design tool is extended for 0.18 μm technology, and design examples from basic block *e.g.* current source/sink to advanced/complex block *e.g.* a three stage operational amplifier for 0.18 μm technology are presented.

Chapter 6 provides a discussion on the thesis' contributions as well as possible future extension of the work.

Finally, four appendices are included for better reader comprehension. Appendix A provides passive domain examples of modeling algorithms based on the binning concept. Appendix B contains passive domain example of the neural modeling approach based on a correction model. Appendix C illustrates the common source amplifier module of the developed design tool through snapshots. Appendix D provides the sample code of the common source amplifier module.

CHAPTER 2

OVERVIEW OF MOSFET MODELING AND DESIGN

2.1 INTRODUCTION

As discussed earlier, MOSFET is more often than not the basic building block on which performance of the entire circuit depends. As such, MOSFET modeling is crucial. In this chapter, top-to-bottom approach is adopted for explaining the MOSFET modeling concept properly. Starting with the complete system *i.e.* design automation; first a brief summary of the analog design approaches used in the design automation is presented. Moving on to the basic building block *i.e.* MOSFET, significance of MOSFET modeling for the design of analog IC is discussed, and some of the currently employed MOSFET

models for the design/analysis of circuits containing MOSFETs are reviewed. Limitations of these models are put forward, and possible alternate models (*e.g.* neural networks) that can potentially address those limitations are reviewed. It should be understood that when a device or circuit being modeled is complex and/or the model being developed is expected to cover a wider input parameter space, two or more modeling techniques can be combined [5] to meet the desired objective.

2.2 Analog Design Approaches

Progress, in terms of the technological advancements, in the analog domain has been substantial. However, research in the analog design automation has been relatively slow. Consequently, not many new tools are developed to aid the analog designers in modeling and designing state of the art analog circuits [6]. This section provides a brief summary of the design approaches, generally employed in commercial and in-house CAD tools.

2.2.1 Optimization Based Design Approach

One of the commonly used analog design approaches was optimization based. In such an approach, sizing of a transistor (*i.e.* geometry of a transistor) for a user-specified circuit topology is formulated as an optimization problem. This concept is reported to be adapted in DELIGHT.SPICE [7], ECSTACY [8], and ADOPT [9]. Transistor sizes are adjusted in an iterative fashion, to satisfactorily meet the user-specifications, employing various optimization tools (e.g. Newton Raphson, Quassi Newton, steepest descent, *etc*). The optimization loop is comprised of a simulator that evaluates the circuit performance at the end of each iteration, and an update block.

Optimization based design approaches have the following limitations.

- *Selection of optimization algorithm:* Selection of a good and appropriate optimization algorithm is vital. Poor choice of optimization algorithm can lead to a local minimum.
- *Selection of optimization variables:* Selection of apt optimization variables is crucial in this approach and design expertise is the key to successful completion of this step.
- *Initial values:* Setting initial values of the optimization variables is an important step in this approach. Lack of design experience can lead to a local minimum, thereby making the optimization meaningless.
- *Design experience:* As discussed in the afore-mentioned steps, a designer should have certain design experience not only regarding the circuit under test, but also with the optimization algorithm used, in case of potential convergence problems.
- *Speed*: This approach could be tedious/tiresome in situations where optimization step enters an infinite loop (*e.g.* getting stuck in some local minima).
- *Lack of design expertise and patience:* Design expertise and patience are two of the chief qualities required in the designer and their deficiency makes this design approach difficult/impossible to be employed.

2.2.2 Layout Based Design Approach

This approach borrows its theme from the extensively used standard cell, gate array, and parameterized cells found in the digital domain [10]. This approach is also referred to as semi-custom/bottom-up approach because of the designs being primarily controlled by layout. In analog domain this concept is implemented with the help of numerous pre-designed blocks of various sizes/configurations. However, this design approach suffers a serious drawback in terms of the design flexibility for performance analog circuits. Pre-

designed blocks implemented in this approach can realize only a small number of discrete points whereas the analog circuits have a wide spectrum of continuous space. A high amount of silicon is wasted in the design of these blocks, making them a costly affair.

Standard cells that are pre-designed and laid-out blocks of varying complexity, residing in the database of the design automation system, address the issue of silicon usage[11]-[13]. These cells, although very popular in the digital domain, are not practical in the analog domain owing to the difficulty in maintaining a rich enough library of these cells to accommodate such a wide spectrum of possible applications in analog domain.

Parameterized cells reported in AIDE2 [14] and CONCORDE [15] improve on the issue of flexibility as these cells fully or partly are customized according to the required function. However, these fixed-layout configurations impose/enforce major restrictions in terms of performance of the analog circuits.

2.2.3 Knowledge-Based Design Approach

This approach employs the available knowledge of the circuit/system to design it. As such, this approach offers the maximum flexibility and therefore covers a wide spectrum of the circuit's performance owing to its fully customized design methodology. A circuit grammar, adapting this concept, to generate bipolar operational amplifiers (OPAMPs) is presented in [16]. However, this approach can not be extended to handle transistor sizing in MOS integrated circuits. Also the grammar itself follows certain conventions, and as such can restrict the approach in the design of unconventional designs. Highly popular design topologies among the designers, namely (i) *Hierarchical*, (ii) *Fixed-Topology* and (iii) *Combined Hierarchical and Fixed Topology*, are reviewed briefly below.

(i) Hierarchical Approach: The underlying idea of this approach is to segment the entire circuit into finite distinct blocks. Each of these blocks is assigned a set of specifications so as to satisfactorily meet the desired circuit performance, when are put together. Thissegmentation/partition process is repeated for finite blocks at various hierarchical levels, and number of levels depends on the circuit being designed, and grammar of the design system. This partition is performed with the help of domain knowledge and hence a great deal of domain knowledge is required.

The knowledge is mainly in the form of design equations and heuristics (basic rules that convey circuit performance upon variations in design parameters). Systems designed with this approach have the highest degree of freedom. Hence, a comparatively small architecture library can lead to a large number of different topologies. Systems designed using this approach are easy to extend and maintain, and make better use of the existing design knowledge. This approach is reported to be adopted in tools like PROSAIC [17], BLADES [18], OASYS [19], and An_Com [20].

(ii) Fixed-Topology Approach: This approach employs a sizing method to compute apt sizes (i.e. geometry) of the devices with the given fixed circuit topology. These fixed, un-sized, device level circuit topologies are stored in a knowledge base together with the necessary domain knowledge for dimensioning the devices [21]. The domain knowledge to be stored depends on how the device sizes are computed. This approach is employed in IDAC [22], OPASYN [23], and OAC [24]. This approach takes into account, only the device dimensions as the legitimate design variables, thereby imposing the strictest limits on the design flexibility among the various knowledge-based approaches.

(iii) Combined Hierarchical and Fixed Topology Approach: In this approach, features of hierarchical and fixed topology approaches are combined. The circuit topology is put together in a hierarchical manner whereas the sizing of transistors is done in a fashion similar to that in the fixed topology. Consequently, this approach offers a higher degree of design flexibility. However, systems designed using this approach are not as flexible as the systems designed using the full-custom hierarchical approach. This approach is presented elaborately in ASAIC [25] and CAMP [26][27].

All the above reviewed approaches have their own advantages and are implemented by various commercial and in-house design tools. However, insufficient use of simulators during the design phase, longer design time, lack of accuracy, and storage space for the knowledge database are some of the limitations imposed. As a part of this thesis work, a design tool is developed exploiting advantages offered by hierarchical and fixed topology approaches. New modeling algorithms are proposed to replace highly accurate but CPU intensive simulators and are used optimally to improve the accuracy of the design.

Hierarchical approach offers reusability of design knowledge by breaking down large and complex circuits into smaller building blocks. Building-blocks, extensively used in circuit design, are simple circuit-blocks that carry out fundamental functions. Current source/sink, current mirror, source follower are few classic examples of building-blocks used in numerous circuits. These blocks are generally comprised of some smaller device-blocks commonly referred as task-blocks (see Fig. 2.1). As discussed earlier, MOSFETs can be considered as the task-blocks in analog domain and breaking of a large circuit to task-block generalizes the circuit design process and ensures the optimum knowledge reusability. As such, accurate modeling of MOSFET is crucial and is discussed further.

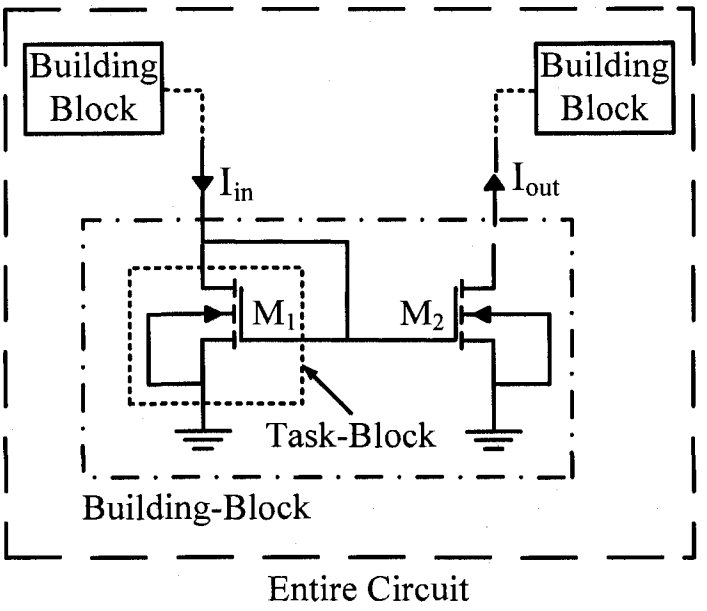


Figure 2.1 Block diagram showcasing the hierarchy in analog circuits.

2.3 Review of MOSFET Modeling Approaches

Device modeling is a bottleneck to design tools; consequently, a robust modeling approach for the modeling purpose is needed. In this section some of the device level (*i.e.* MOSFET level in this case) modeling approaches are reviewed.

2.3.1 Hand Calculation Approach

Commonly used approaches to analog design usually involve some sort of hand calculations. These hand calculations at times can be very lengthy and complex. They are carried out based on assumptions that do not hold true in all the situations. In the hand calculation approach, empirical equations are employed that are solved by the designer to calculate the physical dimensions of the device. Some of the empirical equations used in a first hand calculation approach are listed below.

In the linear/triode region, the Drain Current (I_D) is calculated as [28]

$$I_D = k' \frac{W}{L} \left[(V_{GS} - V_T) V_{DS} - \frac{V_{DS}^2}{2} \right] \tag{2.1}$$

In the Drain in saturation region, the Current (I_D) is calculated as

$$I_D = \frac{k'}{2} \frac{W}{L} (V_{GS} - V_T)^2 (1 + \lambda V_{DS}) \tag{2.2}$$

In the velocity saturated region, the Drain Current (I_D) is calculated as [29]

$$I_D = \frac{1}{1 + \left(\frac{V_{DS}}{E_C L} \right)} k' \frac{W}{L} \left[(V_{GS} - V_T) V_{DS} - \frac{V_{DS}^2}{2} \right] (1 + \lambda V_{DS}) \tag{2.3}$$

In the sub-threshold region, the Drain Current (I_D) is calculated as

$$I_D = I_S e^{\frac{V_{GS}}{\eta kT/q}} \left(1 - e^{\frac{V_{DS}}{\eta kT/q}} \right) \tag{2.4}$$

The Threshold Voltage (V_T) is given by

$$V_T = V_{T0} + \gamma \left(\sqrt{|-2\varphi_F + V_{SB}|} - \sqrt{|-2\varphi_F|} \right), \tag{2.5}$$

and

$$k' = \mu_n \times C_{OX} \tag{2.6}$$

where,

W : Gate width of the transistor.

L : Effective gate length of the transistor.

V_{GS} : Gate source voltage.

V_T : Threshold voltage.

V_{DS} : Drain source voltage.

λ : Body effect parameter.

E_C : Critical Field.

I_S : Current in strong inversion region.

η : Sub-threshold slope factor.

kT/q : Thermal voltage.

V_{T0} : Zero bias threshold voltage.

γ : Body effect constant.

$2\varphi_F$: Surface potential parameter.

V_{SB} : Source bulk voltage.

μ_n : Charge-carrier mobility.

C_{ox} : Gate capacitance across the oxide per unit area.

These hand calculations are based on first-hand knowledge of the device, and could fail for the modern sub-micron CMOS technology. For instance, in large MOSFETs (*i.e.* MOSFETs having larger L), the classical "square-law" current-voltage (I-V) is valid for transistors operating in strong inversion and saturation, while a simple exponential I-V relation works well in sub-threshold region (*i.e.* weak inversion). In the extreme short channel limit, the "square-law" becomes linear and also loses its $1/L$ dependence.

Apart from these cases, accurate hand calculation methods are not available for the MOSFET. One can use $V_{DS} - g_{ds}$ trade-off through the simulation to resolve the problem. A study of various analog design texts shows that this problem is overcome largely by ignoring it. Use of these assumptions has tended not to be fatal, since there has been a considerable margin for error. However, in modern processes, there is little margin for such error. For instance, consider a simple cascode circuit with two transistors and a load. Tolerance for voltage margins in the calculation of V_{Dsat} (drain saturation voltage) is small, since the "voltage budget" across these three elements is very tight. Designers are

forced to use lower DC gate voltages to keep the transistors biased in saturation. This forces transistors to be biased in moderate inversion (rather than strong inversion), a region where there are no good hand calculation formulae. Hence, a more modern and coherent approach for the design and analysis of MOSFET based circuits is required.

2.3.2 Trial and Error Approach

Trial and error is another popular approach used in analog design. This approach is explained with the help of the flowchart depicted in Fig 2.2. In this approach, first, device dimensions are initialized and simulator is employed to check the circuit performance. Values are updated iteratively until the circuit response obtained from the simulator block satisfactorily meets the given user specifications. The simulator block used ensures more accurate design of the circuit/system.

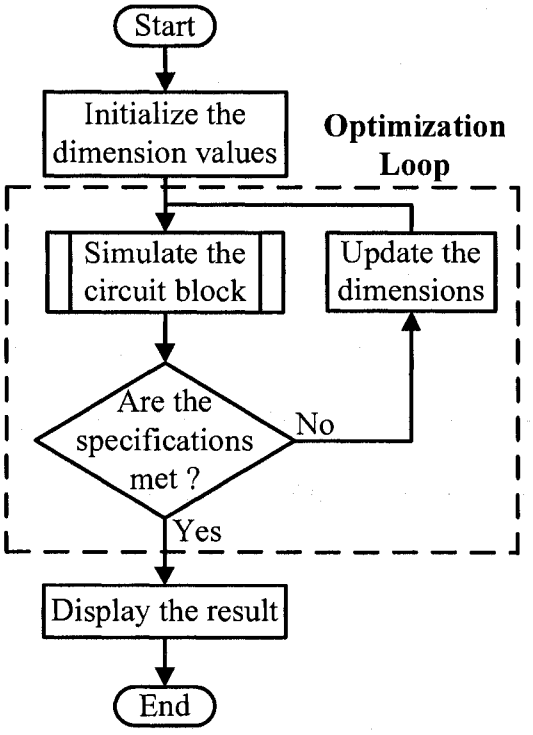


Figure 2.2 Flowchart of the trial and error approach used in the first hand analysis of transistorized circuits.

From the flowchart shown in Fig 2.1, it is obvious that the process is iterative in nature. As in any iterative process, modifications are made at each iteration. Should these modifications be improperly done, the approach can turn into a not so useful infinite loop. In simpler words it can be said that, this process demands experience from a designer so as to ensure proper transitions from one iteration to the next. Once again, the problem lies in the shortage of experienced designers.

This process is fairly tedious, and can be highly frustrating for a designer who does not have good intuition. Even if a designer has experience, achieving correct results in a short time may not always be possible, and finally, since the approach involves manual intervention, it is slower as compared to the automated processes.

2.3.3 Motivation

On one hand, there are several existing modeling approaches that are simple but each with their own limitations. One common disadvantage of all the discussed systems lies in the insufficient use of circuit simulators. Circuits designed from the knowledge (based on available equations) do not necessarily meet user specification, since the equations used to design the circuit usually suffer in terms of accuracy. Few systems incorporate circuit simulators in a loop to design a circuit by iteratively sizing various devices employing numerical algorithms or expertise/heuristics. These systems have potential to produce more efficient designs; however, these systems suffer in terms of longer design time owing to numerous iterations, and hence limit the circuit design exploration.

On a positive side, there are some physics based device simulators that simulate a device accurately but can be cumbersome (*e.g.* Minimos) and accurate models but with too many parameters (*e.g.* BSIM3). It would be nice to develop compact models using

accurate data from such simulators. In the following sections, neural network modeling approach is discussed that has a potential to achieve the accurate compact models for the data obtained from device/circuit simulators.

2.4 NEURAL NETWORK MODELS

2.4.1 Introduction

Neural networks, also called artificial neural networks (ANNs), are the information processing systems with their design inspired by studies of the ability of the human brain to learn from observations and to generalize by abstraction [30]. The very fact that neural networks can be trained to learn any arbitrary nonlinear input–output relationships from corresponding data has resulted in their use in areas such as pattern recognition, speech processing, control, biomedical engineering, *etc* [31]. ANNs have been applied to the modeling of semiconductor devices, circuits and their fabrication processes as well.

Neural networks are first trained to model the electrical behavior of active/passive devices/components. These trained neural networks, often referred to as neural-network models (or simply neural models), can then be used in high-level simulation and design, providing fast answers to the task they have learned. ANNs are efficient alternatives to conventional methods like numerical modeling methods, which could be computationally expensive, or analytical methods, which could be difficult to obtain for new devices, or empirical models, whose range and/or accuracy could be limited [32][33].

A neural network is a set of mathematical equations representing a physical model, relating its output vector $\boldsymbol{y}$ to its input vector $\boldsymbol{x}$. The neural model can then be stated by

$$\boldsymbol{y} = \boldsymbol{f}_{\mathbf{ann}}(\boldsymbol{x}, \boldsymbol{w}), \tag{2.7}$$

where f_{ann} is the neural network and w represents a vector of model parameters. Hence, as with any other mathematical modeling techniques, neural networks can also be used to model laboratory data sets. The objective of the neural network approach is to determine w by a "learning process" using the given input-output laboratory data set. Once the w is determined, the neural model can be used to simulate the phenomenon represented by the given data. One of the main advantages of ANNs is that the output and the input vectors (y and x) can be multidimensional. Parallel processing capability, simple calculations and wide applications are a few other notable advantages. ANNs are generic and have a wider range of applications. In the next section, one of the most commonly used ANNs called the multilayer perceptrons (MLP) network is introduced.

2.4.2 Multilayer Perceptrons (MLP)

Multilayer perceptrons or MLP are the most commonly used neural networks owing to simplicity in terms of their structure and ease in terms of their training. It consists of n number of layers, 1^{st} and n^{th} layers are input and output layers respectively and layers from 2 to n-1 are hidden layers. In this work 3-layer MLP that consists of 1^{st}, 2^{nd}, and 3^{rd} layers as input layer, hidden layer, and output layer respectively is employed. A MLP network consists of two parameters namely, nodes or neurons and links connecting the nodes or neurons.

A 3-layer MLP network with n input neurons, l hidden neurons and m output neurons is depicted in Fig. 2.2. The number of neurons in the input and output layer are fixed according to the problem definition but the number of neurons in the hidden layer can vary. More neurons in hidden layer may result in overlearning and fewer neurons may result in underlearning. For a given modeling problem, deciding the number of neurons in

hidden layers remains an open question. As discussed above, f_{ann} is a set of mathematical equations, representing the MLP network itself and processes faster than any other network. Model parameter vector $\boldsymbol{w}$ is determined and neural network (f_{ann}) acts as a neural model with input $\boldsymbol{x}$ and output $\boldsymbol{y}$. The process of computing the outputs of the neural network starting from its inputs is referred to as "feed-forward computation".

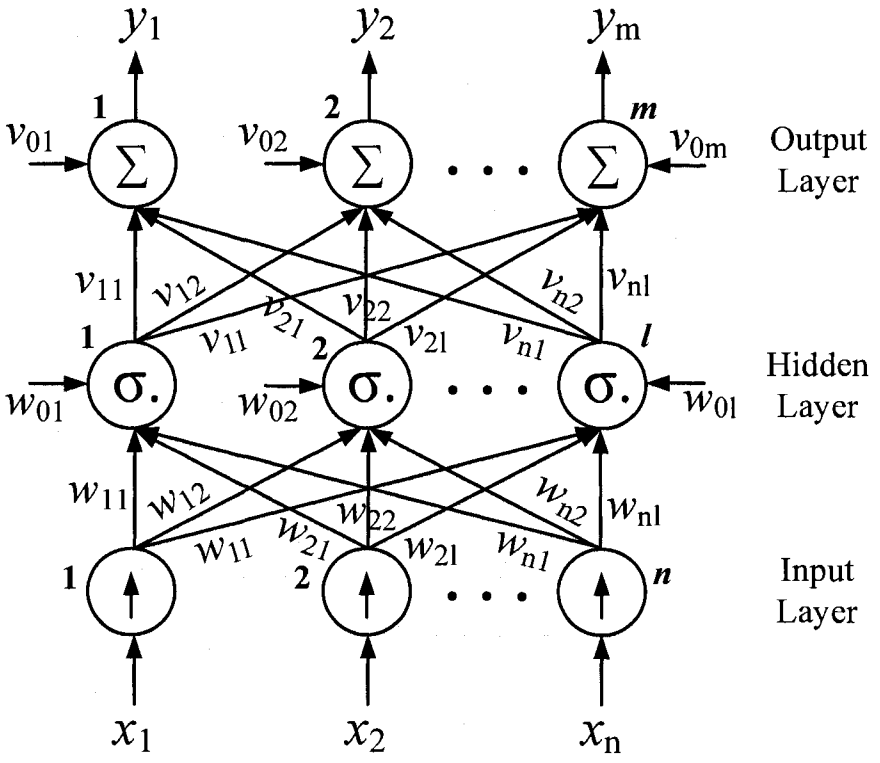


Figure 2.3 Multilayer perceptron (MLP) network.

A. Definition of Parameters

$\boldsymbol{x} = [x_1\ x_2\ \ldots\ x_n]^T$ is the input vector.

$\boldsymbol{y} = [y_1 y_2 \ldots y_m]^T$ is the output vector.

w_{0h} is the bias parameter of h^{th} hidden neuron.

v_{0j} is the bias parameter of the j^{th} output neuron.

w_{gh}, $1 \leq g \leq n$, $1 \leq h \leq l$ is the weight parameter of the link between g^{th} input and h^{th} hidden neuron.

v_{hj}, $1 \leq j \leq m$ is the weight parameter of the link between h^{th} hidden and j^{th} output neuron.

The weight parameter vector w consists of all the weight parameters of the links and the bias parameters (*i.e.* a total of $[(n \times l) + (m \times l) + m + l]^T$ elements). Order of the elements is not vital as it remains same in the updated weight matrix too.

B. Activation Functions

Calculation at different layers is not same. Each neuron in the neural network has an activation function that processes its input to produce an output. Activation function for neuron in each layer is different (represented by ↑ for input layer, **σ.** for hidden layer and ∑ for output layer) as shown in Fig. 2.2. A neuron belonging to the input layer acts as a relay neuron producing an output equivalent to its input. For hidden layer neurons, there are a variety of activation functions like sigmoid, arctangent, hyperbolic tangent *etc.* Commonly used sigmoid function is chosen for this work. The activation function for the neuron in the output layer is summation function. As such, the output of an output neuron is just the weighted sum of its inputs. In case, total input to some neuron in hidden layer or output layer is zero, the output will also be zero. To avoid this situation, an extra weight is attached to each neuron in hidden layer and in output layer which is called bias. Consequently, even if input to that neuron is zero the output is not zero.

C. Feed-forward Computation

(i) Input neuron acts as a relay neuron and there is no calculation/processing. at neurons in input layer.

(ii) The processing by the hidden layer neuron depends on its activation function. The output of the h^{th} hidden neuron with a sigmoid activation function is given by

$$z_{\mathrm{h}} = \frac{1}{1+e^{-\gamma_{\mathrm{h}}}} \tag{2.8}$$

and

$$\gamma_{\mathrm{h}} = w_{0\mathrm{h}} + \sum_{g=1}^{n} w_{g\mathrm{h}} x_{\mathrm{g}}. \tag{2.9}$$

(iii) For output layer neurons processing/calculations involve simple summation function. Thus, output of the j^{th} output neuron is the weighted sum of its inputs given by

$$y_{\mathrm{j}} = v_{0\mathrm{j}} + \sum_{h=1}^{l} v_{h\mathrm{j}} z_{\mathrm{h}}. \tag{2.10}$$

When neural network is employed for the optimization task, derivative information of parameters of the network is required to decide, by how much these parameters are to be modified and in which direction.

D. Derivative Computation

Let's consider a sample calculation for output y_1. Output at y_1 is given by

$$y_1 = v_{01} + v_{11} z_1 + v_{21} z_2 + \ldots + v_{l1} z_1. \tag{2.11}$$

(i) Derivative of y_1 *w.r.t.* bias of the output neuron (*i.e.* v_{01}) can be estimated as

$$\frac{\partial y_1}{\partial v_{01}} = 1. \tag{2.12}$$

In general, (2.10) can be written as $1 \leq j \leq m$,

$$\frac{\partial y_{\mathrm{j}}}{\partial v_{0\mathrm{j}}} = 1. \tag{2.13}$$

(ii) Derivative of y_1 *w.r.t.* weight parameters (*i.e.* links) between hidden layer and output layer can be estimated as

$$\frac{\partial y_1}{\partial v_{11}} = 1; \quad \frac{\partial y_1}{\partial v_{11}} = 1; \quad \frac{\partial y_1}{\partial v_{11}} = 1. \tag{2.14}$$

In general, (2.12) can be written as $1 \le h \le l$,

$$\frac{\partial y_j}{\partial v_{hj}} = z_h. \tag{2.15}$$

(iii) Derivative of y_1 *w.r.t.* bias of the hidden layer neuron (*i.e.* w_{01}) involve following calculations. Chain rule needs to be implemented for this computation which is given by

$$\frac{\partial y_1}{\partial w_{01}} = \frac{\partial y_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial \gamma_1} \cdot \frac{\partial \gamma_1}{\partial w_{01}}. \tag{2.16}$$

From (2.7), (2.8), and (2.10)

$$\frac{\partial z_1}{\partial \gamma_1} = z_1(1 - z_1); \quad \frac{\partial \gamma_1}{\partial w_{01}} = 1; \quad \frac{\partial y_1}{\partial z_1} = v_{11}. \tag{2.17}$$

From (2.14) and (2.15)

$$\frac{\partial y_1}{\partial w_{01}} = v_{11} \cdot z_1(1 - z_1). \tag{2.18}$$

In general, (2.16) can be written as

$$\frac{\partial y_j}{\partial w_{0h}} = v_{hj} \cdot z_h(1 - z_h). \tag{2.19}$$

(iv) Finally, derivative of y_1 *w.r.t.* weight parameters (*i.e.* links) between input layer and hidden layer can be estimated using chain rule as

$$\frac{\partial y_1}{\partial w_{11}} = \frac{\partial y_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial \gamma_1} \cdot \frac{\partial \gamma_1}{\partial w_{11}}. \tag{2.20}$$

From (2.15) and (2.8)

$$\frac{\partial z_1}{\partial \gamma_1} = z_1(1-z_1);\ \frac{\partial \gamma_1}{\partial w_{01}} = x_1;\ \frac{\partial y_1}{\partial z_1} = v_{11}. \tag{2.21}$$

From (2.18) and (2.19)

$$\frac{\partial y_1}{\partial w_{11}} = v_{11} \cdot z_1(1-z_1) \cdot x_1. \tag{2.22}$$

In general, (2.20) can be written as

$$\frac{\partial y_j}{\partial w_{gh}} = v_{hj} \cdot z_h(1-z_h) \cdot x_n. \tag{2.23}$$

This concludes the discussion about the derivative computation of various parameters of the network. The next section describes use of these derivatives in optimization methods. Widely used conjugate gradient method is selected for this work. Optimization in neural network is done by training them with sampled data. Training is of two types (i) sample training where network is trained with one sample data, and (ii) batch training where set of sample data is used to train the network. Batch training is selected for this work.

E. Conjugate Gradient Method

This method is employed for optimization which is called training in neural networks. This method simply calculates values of weight vector $\boldsymbol{w}$ for which $\boldsymbol{f}_{\mathbf{ann}}$ defined in (2.6) closely represents the original problem behavior. First of all, an error function is defined.

$$E = \frac{1}{2}\sum_{k=1}^{p}\sum_{j=1}^{m}(d_{jk} - f_{ann,\,j}(\boldsymbol{x}_k, \boldsymbol{w}))^2. \tag{2.24}$$

Derivative of E *w.r.t.* weight vector $\boldsymbol{w}$ is estimated using chain rule as

$$\frac{\partial E}{\partial \boldsymbol{w}} = \frac{\partial E}{\partial \boldsymbol{y}} \cdot \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{w}}. \tag{2.25}$$

From (2.22)

$$\frac{\partial E}{\partial \boldsymbol{y}} = \sum_{k=1}^{p} \sum_{j=1}^{m} (f_{\text{ann, j}}(\boldsymbol{x}_{\text{k}}, \boldsymbol{w}) - d_{\text{jk}}). \tag{2.26}$$

Calculation of derivatives of $\boldsymbol{y}$ *w.r.t.* each parameter of vector $\boldsymbol{w}$ is described in detail in the derivative computation section. Conjugate gradient method can then be implemented to minimize error E to zero by adjusting vector $\boldsymbol{w}$, and is described in following steps:

Step #1 : Weight vector $\boldsymbol{w}$ of the size $[(n \times l) + (m \times l) + m + l]^T$ is constructed and initialized. Order of the vector elements is not crucial as it remains same for updated weight vector $\boldsymbol{w}$.

Step #2 : All the required derivatives are calculated as discussed in previously.

Step #3 : $\frac{\partial E}{\partial \boldsymbol{w}}$ is calculated using previously obtained derivative information and (2.23).

Step #4 : Weight vector $\boldsymbol{w}$ is modified using following formulae.

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w}_{\text{old}} + \eta \boldsymbol{d} \tag{2.27}$$

and

$$\boldsymbol{d} = -\boldsymbol{g} \tag{2.28}$$

and

$$\boldsymbol{g} = \frac{\partial E}{\partial \boldsymbol{w}} \Big| \boldsymbol{w} = \boldsymbol{w}_{\text{old}}, \tag{2.29}$$

where η is called learning rate and is generally considered to be small, say 0.1, to avoid cancellation of present and past values of weights, $\boldsymbol{d}$ is successive update direction vector and $\boldsymbol{g}$ is gradient vector.

2.5 NEED FOR NEW MODELING ALGORITHMS

When a device/component being modeled is highly nonlinear and/or when the model being developed is expected to cover a wider input parameter space, advanced ANNs utilizing knowledge (*e.g.* knowledge based neural networks or KBNN) are employed [33][34]. Alternatively, two or more existing modeling techniques can be integrated [5]. However, such approaches lead to increased model complexity. Besides, such advanced approaches require the users to have an in-depth understanding of ANN concepts for them to be able to develop satisfactorily accurate models. Thus new modeling algorithms are needed to enhance the modeling process and to make it simple for the novice users.

2.6 SUMMARY

In this chapter, some basic concepts regarding the electrical (*i.e.*, I-V relation) model of a MOSFET along with analog design approaches in practice have been presented. Motivation for the work owing to shortcomings of current approaches has been briefly discussed. The concept of neural networks is briefly introduced along with the most commonly used neural models, and the need for new modeling algorithms is elucidated. In the following few chapters the goal of modeling is pursued and hence, novel modeling algorithms are proposed that offer substantial advantage towards accuracy and efficiency as regards the computational resources are concerned.

CHAPTER 3

MOSFET MODELING BASED ON BINNING ALGORITHMS

3.1 INTRODUCTION

Neural approaches to developing device/component models have been described in [29]-[31]. As discussed in chapter 2 (section 2.5), when a 3-layer MLP network fails to model the given device accurately, additional hidden layer neurons or additional hidden layers can be added. A recent trend is to employ the existing knowledge to architect advanced structures, known as Knowledge Based Neural Networks (KBNN), described in

[32][33]. For a given model accuracy, KBNN have been shown to reduce the need for training data. However, such advanced approaches require the users to have an in-depth understanding of ANN/KBNN concepts, to be able to develop satisfactorily accurate models. Motivated by this, the goal is to keep the model structure simple, while attaining satisfactory model accuracies.

In this chapter, new device modeling algorithms are proposed, which allow accurate device modeling over a wider input parameter space. Starting from a given training data set, proposed algorithms employ simple 3-layer MLP structures and generate a set of sub-models. Each of these models represents the device behaviors in a subspace of the overall input space. The sub-models are interfaced to generate an overall model that satisfactorily meets the given accuracy specification. Illustration examples confirming the validity of both the algorithms in active as well as passive domain are developed. Since examples of passive component modeling are not part of the thesis framework, they are included in appendix A.

3.2 STANDARD NEURAL MODELING APPROACH

Let $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ represent a vector of inputs or design parameters, and y represent response of the device/component being modeled. The objective of modeling is to determine a relationship f, *e.g.* neural network, such that

$$y = f(\boldsymbol{x}). \tag{3.1}$$

In standard neural modeling, a 3-layer MLP is chosen to deduce f. Let k represent the number of available training data of the form $(\boldsymbol{x}^i, y^i)$, where $i = 1, 2, \ldots, k$. Let f_{ann} represent the trained neural model. A percentage average error E of the neural model can

be calculated as,

$$E = \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(\frac{y^{i} - f_{\text{ann}}(\boldsymbol{x}^{i})}{y^{i}}\right)^{2}} \times 100. \tag{3.2}$$

In situations, where data is expensive/scarce, training data may also be used for model validation, although not advisable. For simplifying the discussion, E is assumed to represent the model accuracy. Assuming that the output y is highly sensitive to one of the inputs, say x_2, it could be difficult/impossible to deduce f_{ann} that satisfies user-specified accuracy. One possibility, as discussed earlier, is to add more hidden neurons or layers, but this may not always work. Advanced structures, *e.g.* KBNN, can also be employed; however, such an approach requires an in-depth understanding of ANN/KBNN concepts.

3.3 Binning Algorithm for Single Dimensional Models

Binning/partitioning means dividing the model input parameter space into subspaces and developing corresponding sub-models that collectively span the entire input space of interest. In those challenging situations mentioned above, where the standard algorithm fails to generate a neural model with a lower or satisfactory E, the highly nonlinear input parameter, say x_2, is removed from vector $\boldsymbol{x}$. The reduced input space is then divided into finite intervals using uniform-grids along the x_2 axis. A model $f_{\text{ann, j}}$ is deduced for each of these intervals, where j denotes the j^{th} interval. Parameter x_2 is referred to as the binning parameter, and the process is referred to as binning. Given a modeling problem, the challenge is to identify the binning parameter. Based on the device being modeled, there can be two cases: (i) Sensitivity analysis of the device is possible. (ii) Sensitivity analysis

is impossible (*i.e.* empirical equations are unavailable) or it is impractical (*i.e.* available equations are CPU-intensive).

Consider case (i), where simple first-hand analysis equations of form (3.1), for the component being modeled, are available but do not hold true for the latest technology. Derivatives of output y corresponding to each of the inputs *i.e.* $\partial y/\partial x_1$, $\partial y/\partial x_2$, ..., $\partial y/\partial x_n$ are first computed over the entire space of interest. The resulting sensitivity information is analyzed and the input that highly affects the output y is chosen as the binning parameter (*BP*). If the output is equally sensitive to two or more inputs, the input with the least range is selected as the binning parameter. In case (ii), selection of the binning parameter based on a trial-and-error method is proposed. Considering one input at a time, the process of binning is repeated n times. For each of the n potential binning parameters, an aggregate error measure E_{avg} is computed from corresponding sub-models in a manner similar to (2). The candidate parameter leading to the least E_{avg} is selected as the binning parameter. In the trial-and-error step of the proposed algorithm, the number of intervals corresponding to each of the binning parameters is chosen to be small.

In either case, once the binning parameter is identified, it is removed from the input vector $\boldsymbol{x}$ and the remaining subspace is divided into finite intervals using uniform-grids along the binning parameter. In this sub-model development step, a relatively larger number of intervals are advocated. A 3-layer MLP sub-model is developed for each of these intervals and these sub-models are interfaced to generate an accurate overall model. A flow-chart of the proposed single dimensional binning algorithm is depicted in Fig. 3.1. The overall model generated using the proposed algorithm is depicted in Fig. 3.2. As can be seen, the value of the binning parameter helps select an appropriate sub-model during

the utilization of the model.

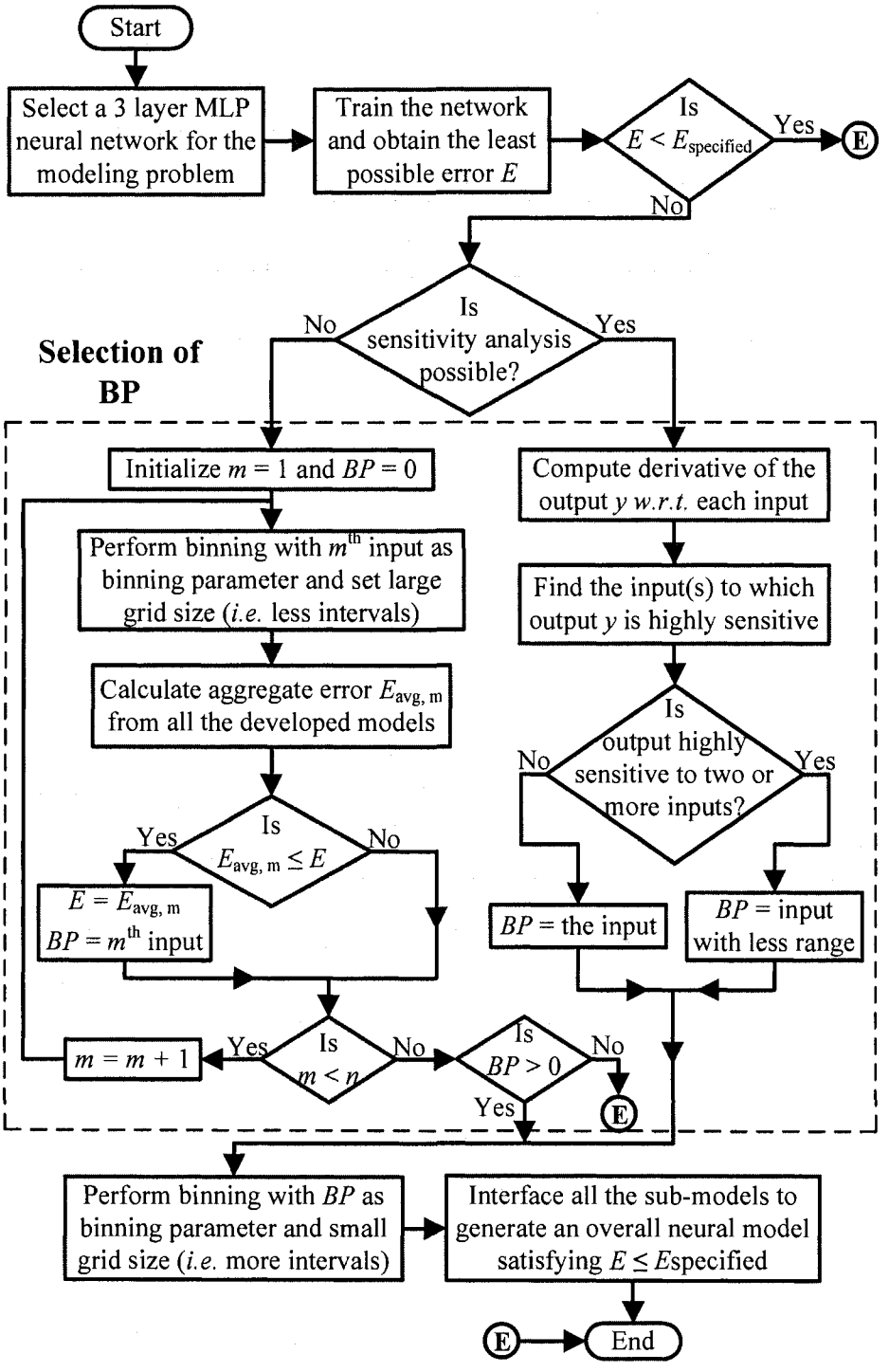


Figure 3.1 Flow-chart illustrating the binning algorithm for single dimensional problems.

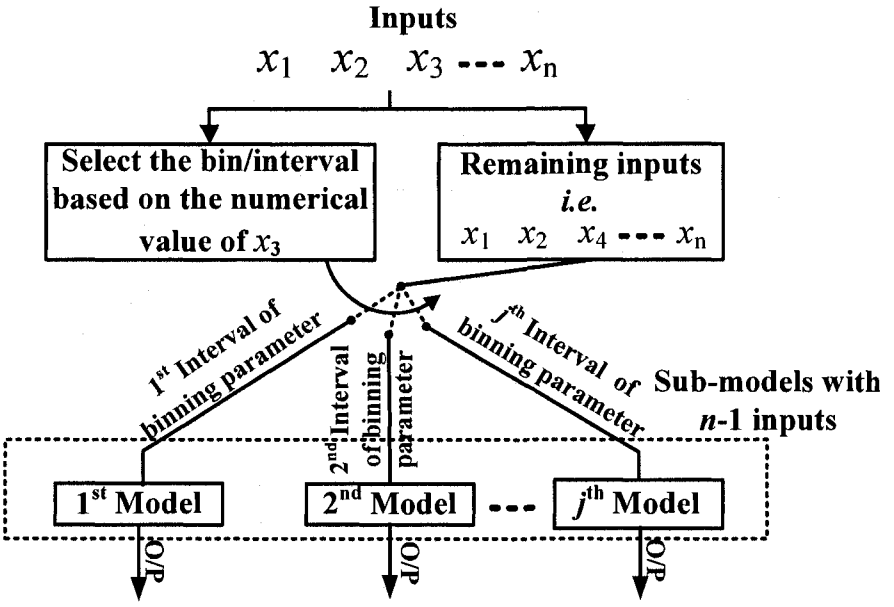


Figure 3.2 Overall model developed using the proposed binning algorithm (assuming x_3 to be the binning parameter).

3.4 BINNING ALGORITHM FOR MULTI-DIMENSIONAL MODELS

In this section, an enhanced modeling algorithm, which is an extension of the binning concept, presented in earlier section, to multi-dimensional modeling, is proposed. Starting from a given training data set, the algorithm employs simple 3-layer MLP structures and generates a set of sub-models similar to the earlier proposed algorithm. The sub-models are interfaced to generate an overall model that meets the given accuracy specification. Division of the parameter space into subspaces is facilitated by the binning concept. In situations where derivative information of the device output *w.r.t.* the input parameters is easy to compute/estimate, such sensitivity information is used to divide the input space. In situations where such analysis is not feasible, computation of impact indices of the inputs over all the outputs is advocated. In addition, standard deviation of each input is evaluated, and binning is performed using such statistical information.

As mentioned earlier, based on the device being modeled, there can be two cases. (i) Sensitivity analysis of the device/component is possible, (ii) Sensitivity analysis is impossible (*i.e.* unavailability of empirical equations) or it is impractical (*i.e.* available equations are CPU-intensive).

Consider case (i), where simple first-hand equations of form (3.1) for the device are available; however, these equations are not satisfactorily accurate. Derivatives of output $\boldsymbol{y}$ *w.r.t.* each of the inputs *i.e.* $\partial y_1/\partial x_1$, $\partial y_2/\partial x_1$, ..., $\partial y_p/\partial x_1$, $\partial y_1/\partial x_2$, $\partial y_2/\partial x_2$, ..., $\partial y_p/\partial x_n$, are derived over the entire input parameter space that leads to an approximate sensitivity analysis. In case (ii), selection of the binning parameter is proposed based on impact indices. For instance, impact index of x_m on y_p is estimated (*via* training data) as

$$g_{mp} = \sqrt{\sum_{i=1}^{s_m} \left[\frac{\left(y_p^{i+1} - y_p^i \right)}{\left(x_m^{i+1} - x_m^i \right)} \right]^2}, \tag{3.3}$$

where s_m is the number of grids along m^{th} input (*i.e.* x_m). Standard deviation (σ_m) of the m^{th} input parameter is also calculated as

$$\sigma_m = \sqrt{\frac{1}{s_m} \sum_{i=1}^{s_m} \left(x_m^i - \overline{x_m} \right)^2}, \tag{3.4}$$

where $\overline{x_m}$ is the mean of x_m over all data samples.

In either case, the input parameter with low deviation and high sensitivity (or impact index) is selected as the binning parameter. It is then removed from vector $\boldsymbol{x}$ and the remaining subspace is divided into a finite number of intervals using uniform-grids along the binning parameter. A 3-layer MLP sub-model is developed for each of these intervals

and these sub-models are interfaced to generate an overall model.

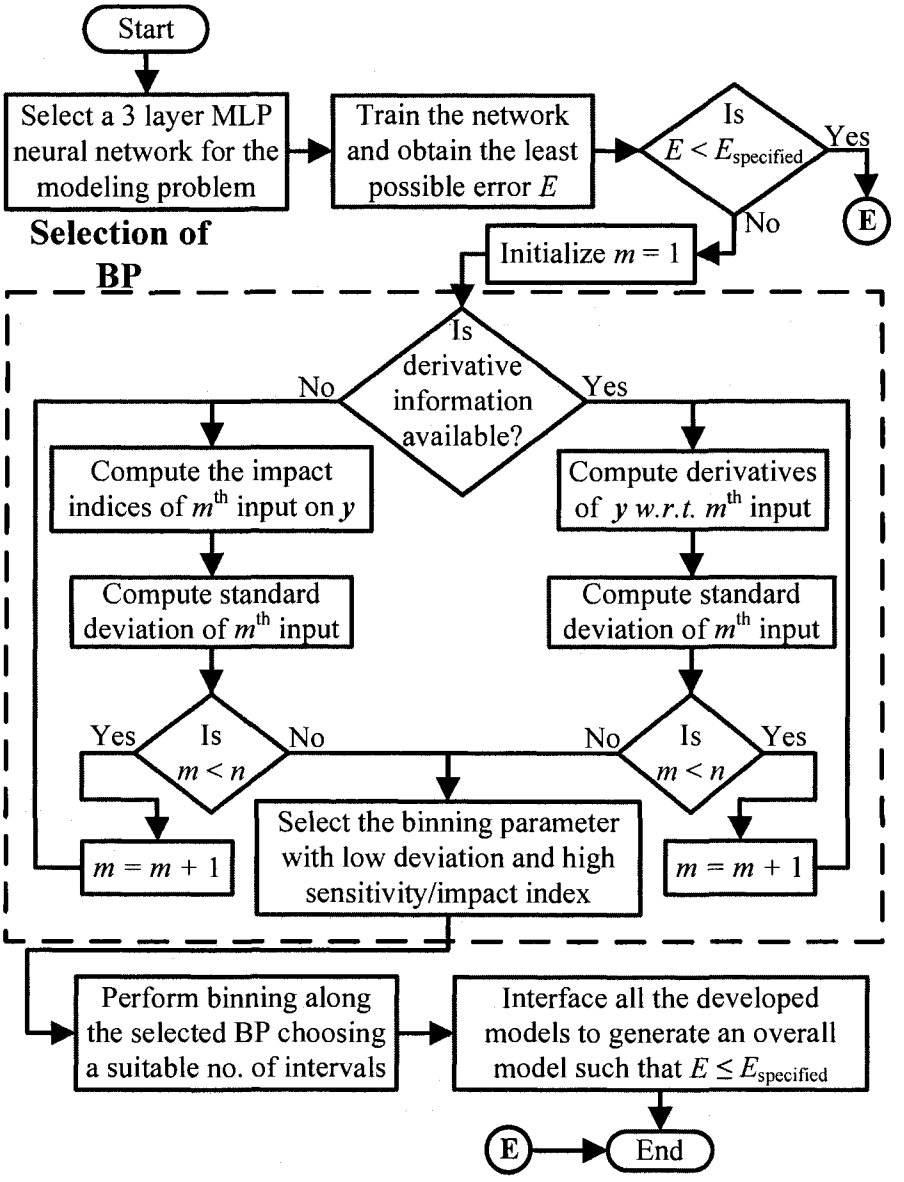


Figure 3.3 Flow-chart of the proposed multi-dimensional binning algorithm.

3.5 ILLUSTRATION EXAMPLES

For both the proposed modeling algorithms, illustrative examples covering active device modeling are presented in this section, while those covering passive component modeling are included in appendix A.

3.5.1 Single Dimensional Modeling Example (MOSFET)

Commercial models (*e.g.* BSIM3) involve numerous model parameters, and model reduction is preferable. The modeling problem can be stated as

$$I_D = f(L, W/L, V_{GS}, V_{DS}), \tag{3.5}$$

where L and W represent gate length and width, V_{GS} and V_{DS} are gate and drain voltages, and I_D represents drain current. Training data for modeling the transistor is obtained using a detailed transistor model from a well-known simulator, namely *HSPICE*. A datasheet of I_D is generated by sweeping the input parameters L, W/L, V_{DS}, and V_{GS} along uniform grids. As an example, the training data corresponding to L = 1μm and V_{GS} = 1.4V is shown in Fig. 3.4(b).

First, a neural network model of the transistor is developed using the standard approach. A 3-layer MLP with four inputs and one output shown in Fig. 3.4(a) is used. *NeuroModeler* [34] is used for training, resulting in a best possible neural model. Model responses corresponding to the training data in Fig. 3.4(b) are shown in Fig. 3.4(c). In an attempt to closely inspect the neural model, model errors for different sub-ranges along the V_{GS} axis are computed (see Table 3.1). "Too high", implies that the error is so high and the neural model fails to emulate the training data. Assuming that the acceptable average error is 5%, the model appears to be good for V_{GS} > 1.2V; however, exhibits

worst-case errors > 38.2%, which is not acceptable.

For the transistor, first-hand analysis equations are known but the equations do not hold true in the submicron region. The standard equation for estimating drain current of a CMOS transistor operating in saturation region is given by

$$I_D = \frac{\mu_n C_{ox} W (V_{GS} - V_T)^2 (1 + \lambda V_{DS})}{2L}, \tag{3.6}$$

where μ_n is electron mobility, C_{ox} is gate-oxide capacitance per unit area, V_T is threshold voltage, and λ represents channel length modulation. As such, the proposed binning algorithm can be applied. In order to identify the binning parameter, sensitivity analysis needs to be performed. In other words, derivatives of the output I_D w.r.t. all the inputs need to be computed. For instance, derivative of I_D w.r.t. V_{GS} can be computed using

$$\frac{\partial I_D}{\partial V_{GS}} = \frac{\mu_n C_{ox} W (V_{GS} - V_T)(1 + \lambda V_{DS})}{L}. \tag{3.7}$$

Based on the numerical values of the derivatives, I_D is observed to be more sensitive to L and V_{GS}, and hence these parameters are chosen as candidate binning parameters. The range of V_{GS} is relatively small for the micro-nano technology (MNT) devices. On the other hand, L is allowed to vary over a relatively wider range. Consequently, V_{GS} is selected as the binning parameter. It is then removed from $\boldsymbol{x}$, and the remaining space is divided into 16 intervals using a uniform-grid along the V_{GS} axis. Neural sub-models corresponding to all 16 intervals are developed and are interfaced resulting in an overall model. Average errors of the model are around 0.2% and worst-case errors are less than 3% along the entire V_{GS} axis. Model responses corresponding to L = 1μm and V_{GS} = 1.4V are shown in Fig 3.4(d).

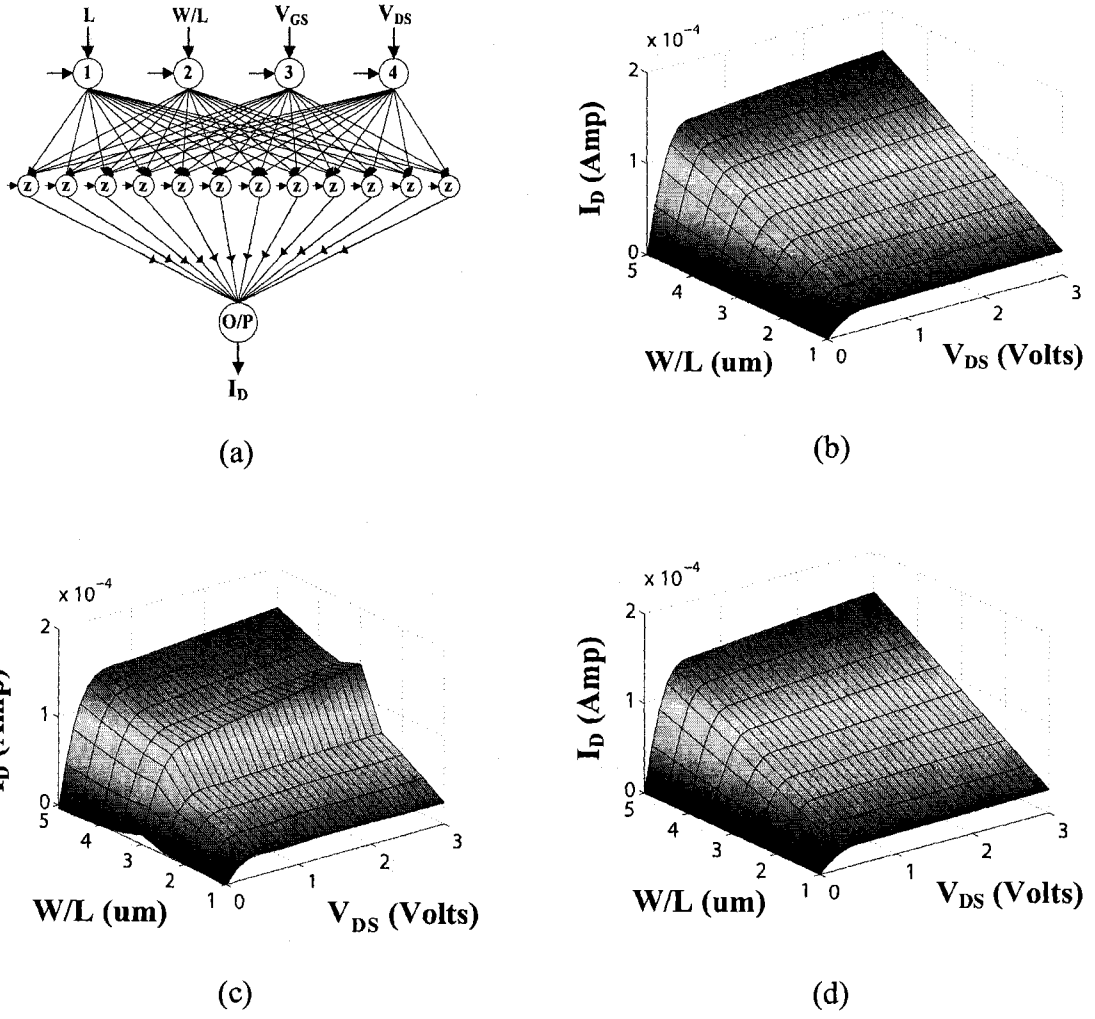


Figure 3.4 (a) Implemented 3-layer MLP model for the standard approach, (b) Training data corresponding to V_{GS} = 1.4V, L = 1μm, (c) Neural Model responses using standard approach, and (d) Neural Model responses using proposed single dimensional binning algorithm.

Table 3.1 COMPARISION OF MODEL ACCURACIES FOR SINGLE DIMENSIONAL MOSFET MODELING EXAMPLE

V_{GS} Range in (V)	Neural Model Using The Standard Approach		Overall Model Using The Proposed Algorithm	
	Average Error	*Worst-Case Error*	*Average Error*	*Worst-Case Error*
0.4-0.5	Too High	Too High	0.23%	2.98%
0.5-0.6	Too High	Too High	0.26%	2.97%
0.6-0.7	Too High	Too High	0.28%	2.92%
0.7-0.8	Too High	Too High	0.27%	2.93%
0.8-0.9	70.53%	1165.50%	0.20%	2.94%
0.9-1.0	25.72%	326.38%	0.22%	2.69%
1.0-1.1	10.90%	124.11%	0.20%	2.90%
1.1-1.2	5.75%	52.84%	0.17%	2.49%
1.2-1.3	3.78%	39.74%	0.20%	2.94%
1.3-1.4	2.61%	38.20%	0.16%	2.67%
1.4-1.5	2.04%	48.25%	0.15%	2.98%
1.5-1.6	1.69%	72.30%	0.15%	2.71%
1.6-1.7	1.38%	86.09%	0.12%	2.64%
1.7-1.8	1.17%	89.99%	0.19%	2.98%

3.5.2 Multi- Dimensional Modeling Example (MOSFET)

This example illustrates proposed multi-dimensional binning algorithm. Parameters that decide gain of a MOS transistor are modeled. The modeling problem can be stated as

$$[g_m, g_{ds}] = f(L, W/L, V_{DS}, I_D), \tag{3.9}$$

where L and W represent gate length and width, V_{DS} is the drain voltage, and I_D represents drain current. Finally, g_m and g_{ds} represent transconductance and output conductance of the MOS transistor respectively. Bulk voltage V_{BS} and gate voltage V_{GS} are kept fixed at -1.5V and 0.9V respectively. Training data for modeling the transistor is obtained using a

detailed transistor model from a well-known simulator, namely *HSPICE*. A datasheet of I_D is generated by varying the inputs L, W/L, and V_{DS} along uniform grids. Furthermore, values of g_m and g_{ds} are derived from the obtained datasheet. As an example, training data for g_{ds} with $L = 1\mu m$ and $V_{GS} = 0.9V$ is shown in Fig. 3.5(b).

First, a neural network model of the transistor is developed using the standard approach. A 3-layer MLP with four inputs and two outputs shown in Fig. 3.5(a) is used. *NeuroModeler* is used for training, resulting in a best possible neural model. Model responses corresponding to the training data in Fig. 3.5(b) are shown in Fig. 3.5(c). In an attempt to closely inspect the neural model, model errors for different sub-ranges along the L axis are computed (see Table 3.2). "Too high", implies that the error is so high and the neural model fails to emulate the training data. Assuming that the acceptable average error is 5%, the model appears to be good for $L < 0.4\mu m$; however, exhibits worst-case errors > 35.89%, which is not acceptable.

The standard equations for estimating g_m and g_{ds} of a CMOS transistor operating in the saturation region are given by

$$g_m = \frac{\mu_n C_{ox} W (V_{GS} - V_T)(1 + \lambda V_{DS})}{L}, \tag{3.10}$$

and

$$g_{ds} = \frac{\mu_n C_{ox} W (V_{GS} - V_T)^2 \lambda}{2L}, \tag{3.11}$$

where μ_n is electron mobility, C_{ox} is gate-oxide capacitance per unit area, and V_T is threshold voltage. As such, the proposed binning algorithm can be applied. In order to identify the binning parameter, sensitivity analysis needs to be performed. In other words,

derivatives of the outputs g_m and g_{ds} *w.r.t.* all the input parameters need to be computed. For instance, derivative of g_m and g_{ds} *w.r.t.* L can be computed using

$$\frac{\partial g_m}{\partial L} = -\frac{\mu_n C_{ox} W (V_{GS} - V_T)(1 + \lambda V_{DS})}{L^2}, \tag{3.12}$$

and

$$\frac{\partial g_{ds}}{\partial L} = -\frac{\mu_n C_{ox} W (V_{GS} - V_T)^2 \lambda}{2L^2}. \tag{3.13}$$

Based on the numerical values of the derivatives, g_m and g_{ds} are both observed to be more sensitive to L and hence L is chosen as the binning parameter. It is then removed from $\boldsymbol{x}$, and the remaining space is divided into 10 intervals using a uniform-grid along the L axis. Neural sub-models for all the 10 intervals are developed and are interfaced resulting in an overall model. Average errors of the model for both outputs are less than 0.8% and worst-case errors are less than 3% along the entire L axis. Model responses corresponding to $L = 1\mu m$ and $V_{GS} = 1.4V$ are shown in Fig 3.5(d).

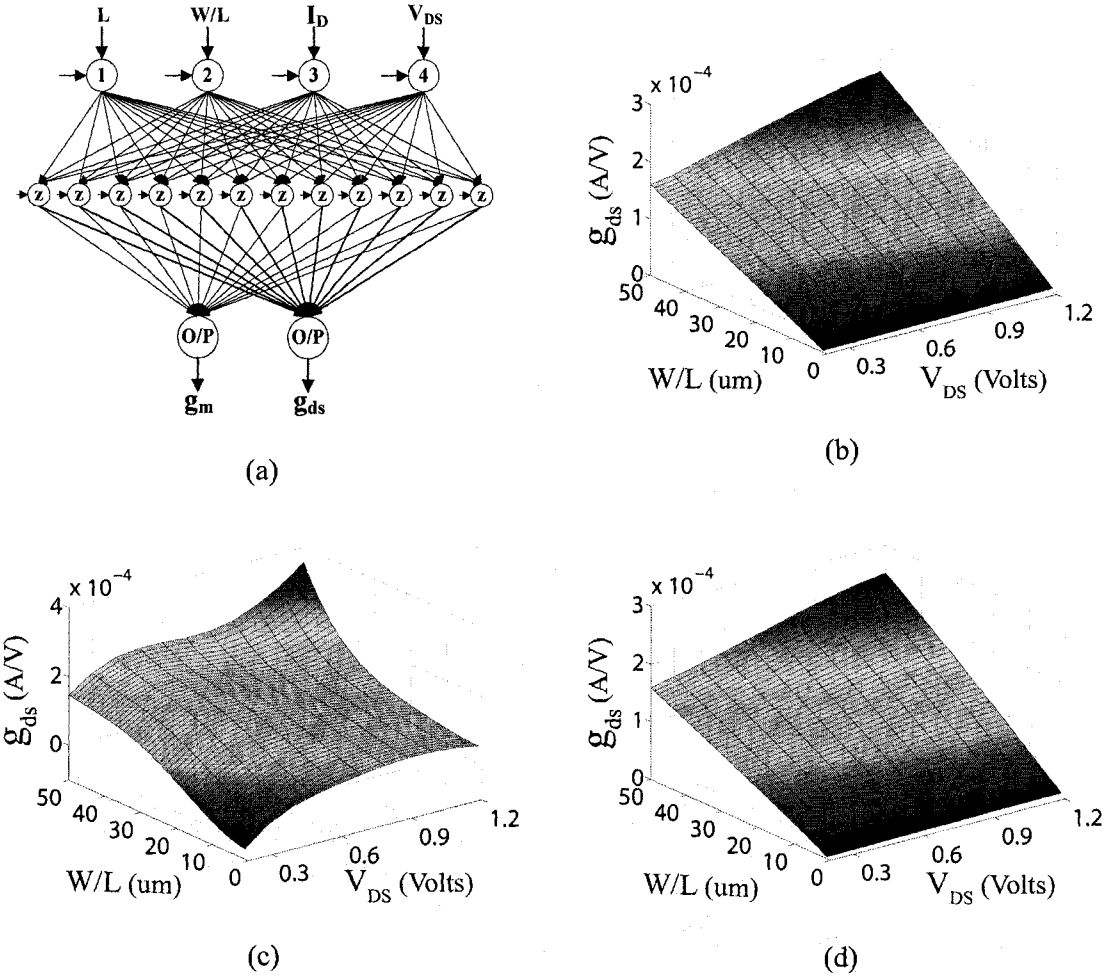


Figure 3.5 (a) Implemented 3-layer MLP model for the standard approach, (b) Training data corresponding to V_{GS} = 1.4V, L = 1μm. (c) Neural Model responses using standard approach, and (d) Neural Model responses using proposed multi-dimensional binning algorithm.

Table 3.2 COMPARISION OF MODEL ACCURACIES FOR THE MUTI-DIMENSIONAL MOSFET MODELING EXAMPLE

L Range in (μm)	**Neural Model Using The Standard Approach**				**Overall Model Using The Proposed Algorithm**			
	Average Error		***Worst-Case Error***		***Average Error***		***Worst-Case Error***	
	g_m	g_{ds}	g_m	g_{ds}	g_m	g_{ds}	g_m	g_{ds}
0.18-0.2	3.75%	1.07%	48.37%	48.84%	0.23%	0.37%	2.34%	2.70%
0.2-0.3	2.17%	1.45%	37.07%	60.63%	0.26%	0.32%	2.84%	2.30%
0.3-0.4	1.35%	3.78%	35.89%	44.21%	0.18%	0.23%	2.12%	2.43%
0.4-0.5	1.63%	19.27%	45.11%	71.87%	0.32%	0.27%	2.36%	2.84%
0.5-0.6	3.72%	7.083%	Too High	Too High	0.27%	0.79%	2.07%	2.75%
0.6-0.7	7.54%	Too High	Too High	Too High	0.19%	0.41%	2.34%	2.61%
0.7-0.8	12.67%	Too High	Too High	Too High	0.14%	0.55%	2.77%	2.93%
0.8-0.9	21.39%	Too High	Too High	Too High	0.21%	0.36%	2.11%	2.54%
0.9-1.0	37.19%	Too High	Too High	Too High	0.28%	0.45%	2.35%	2.88%

3.6 SUMMARY

In this chapter, two new device modeling algorithms based on binning concept have been proposed. While the first algorithm is for single dimensional modeling problems, the second extends the concept to multi-dimensional problems. Both the algorithms begin with selection of the binning parameter. Overall input space is divided into finite intervals along the binning parameter axis, and model for each of the intervals are developed. All these developed models are then interfaced to generate an overall model that meets the user-specified accuracy. Models providing accuracies in between the standard neural models and proposed model do exist. However they require detailed study of the neural network concepts as mentioned earlier. Proposed algorithms eliminate the requirement of in-depth knowledge of the modeling technique being used, by using simple 3-layer MLP

network. Proposed algorithms attain the best possible accuracy compared to other neural network models and make it easier for the user to develop accurate yet simple models for the devices/components. Active device modeling examples are presented in this chapter, while the passive component modeling examples are included in appendix A. Illustration examples show that the proposed algorithms have a potential to eliminate need of advanced/complex structures.

CHAPTER 4

MOSFET MODELING BASED ON A CORRECTION MODEL

4.1 INTRODUCTION

Device/component modeling can be difficult. This could be due to many reasons. One such reason is, output being highly non-linear to one of the inputs. For such situations, new device modeling algorithms have been proposed in chapter 3. However, if the output is highly non-linear to more than one input, identification of the binning parameter could be difficult. Also, the identified binning parameter might not be able to develop accurate

models. As such, in this chapter, a new ANN modeling approach based on a correction model concept is introduced for those challenging situations mentioned above. Correction model is an intermediary model of a relatively higher accuracy that helps enhance the accuracy of the initially/originally less accurate desired model while keeping the model structure simple. Validity of the proposed modeling approach in active as well as passive domain is verified through illustration examples. Since, the example of passive component modeling is not part of the thesis framework it is included in appendix B.

4.2 STANDARD NEURAL MODELING APPROACH

This section revises the concept of standard neural modeling approach and introduces a few new terminologies. Let $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ represent a vector of model inputs and y represent the output of the device being modeled such that

$$y = f(\boldsymbol{x}), \tag{4.1}$$

where f represents the functional relationship (or the training data). In standard neural modeling, a 3-layer MLP is trained to realize a neural model f_{ann} closely representing f. Let k represent the number of available training data of the form (x^i, y^i). Quality measures, *i.e.* average error E and worst-case error E_{worst}, of f_{ann} can be evaluated using

$$E = \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(\frac{y^i - f_{ann}(\boldsymbol{x}^i)}{y^i}\right)^2} \times 100. \tag{4.2}$$

and

$$E_{worst} = \max_i \left(\left|\frac{y^i - f_{ann}(\boldsymbol{x}^i)}{y^i}\right| \times 100\right). \tag{4.3}$$

For ease of the concept illustration, only one of the device outputs are modeled in the examples presented. In cases, where the data is expensive, training data may be used for model validation, although such an approach is not advisable. To simplify the discussion however, E is assumed to denote model accuracy. Assuming that the output y is highly sensitive to one of the inputs, it could be difficult to deduce an MLP network f_{ann}, which satisfies the user-specified average model accuracy E_{user}. In such scenarios, more hidden layers or neurons can be added, but this approach may not always work. Advanced structures, *e.g.* KBNN, can be employed; however, such an approach requires an in-depth understanding of the ANN concepts.

4.3 Proposed ANN Modeling Approach

Consider a scenario, where the standard approach fails to generate f_{ann} satisfying $E \leq E_{user}$. Two neural models, namely the desired model f_{ann} and a correction model $f_{ann,\,c}$ are defined. While the structure of f_{ann} itself is based on the given modeling problem, the structure of $f_{ann,\,c}$ is not known *apriori*. Let

$$y = f_{ann}(x_1, x_2, x_3, \ldots, x_n) \tag{4.4}$$

represent the desired model. For the first time ever, a set of potential correction models are defined as

$$\begin{aligned} x_{1,\,c} &= f_{ann,\,1}(y, x_2, x_3, \ldots, x_n) \\ x_{2,\,c} &= f_{ann,\,2}(x_1, y, x_3, \ldots, x_n), \\ &\vdots \\ x_{n,\,c} &= f_{ann,\,n}(x_1, x_2, x_3, \ldots, y) \end{aligned} \tag{4.5}$$

where $f_{ann,\,i}$ and $x_{i,\,c}$ represent the i^{th} potential correction model and its output respectively. As can be seen, outputs of these correction models are inputs of the desired model. Since the primary focus of this work is to keep both model structure and training as simple as possible, 3-layer MLP networks are used to develop f_{ann} as well as $f_{ann,\,i}$, $1 \le i \le n$.

An important step in the proposed approach is to identify the correction model. Consider a potential correction model $f_{ann,\,1}$. After rearranging training data accordingly, $f_{ann,\,1}$ is trained using *NeuroModeler*. Given a rearranged sample as input, $f_{ann,\,1}$ provides an output $x_{1,\,c}$ (which closely approximates x_1). Quality measure $E_{1,\,c}$ of $f_{ann,\,1}$ is then evaluated. This process is repeated n times, resulting in n potential correction models with corresponding error measures $E_{1,\,c}$, $E_{2,\,c}$, …, $E_{n,\,c}$ respectively. The correction model $f_{ann,\,j}$, is then selected using

$$j = arg\left(\min_i E_{i,\,c}\right). \tag{4.6}$$

It is important to note that the earlier developed f_{ann} can not be used as a stand-alone model owing to its unacceptable quality. As elaborated in the following pseudocode, the correction model $f_{ann,\,j}$ is employed (see Fig. 4.1) to enhance the accuracy of f_{ann} leading to the proposed approach (see Fig. 4.2).

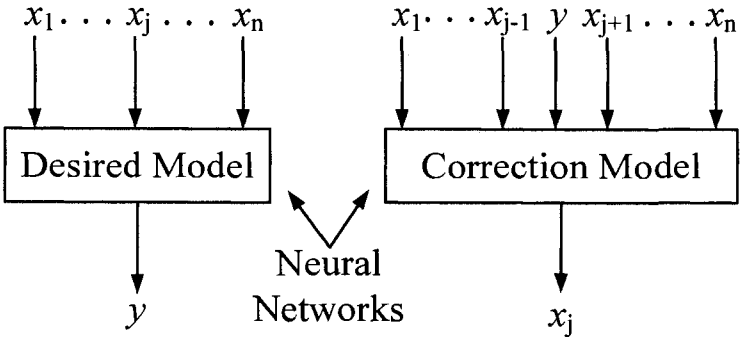


Figure 4.1 Pictorial depiction of desired and correction models.

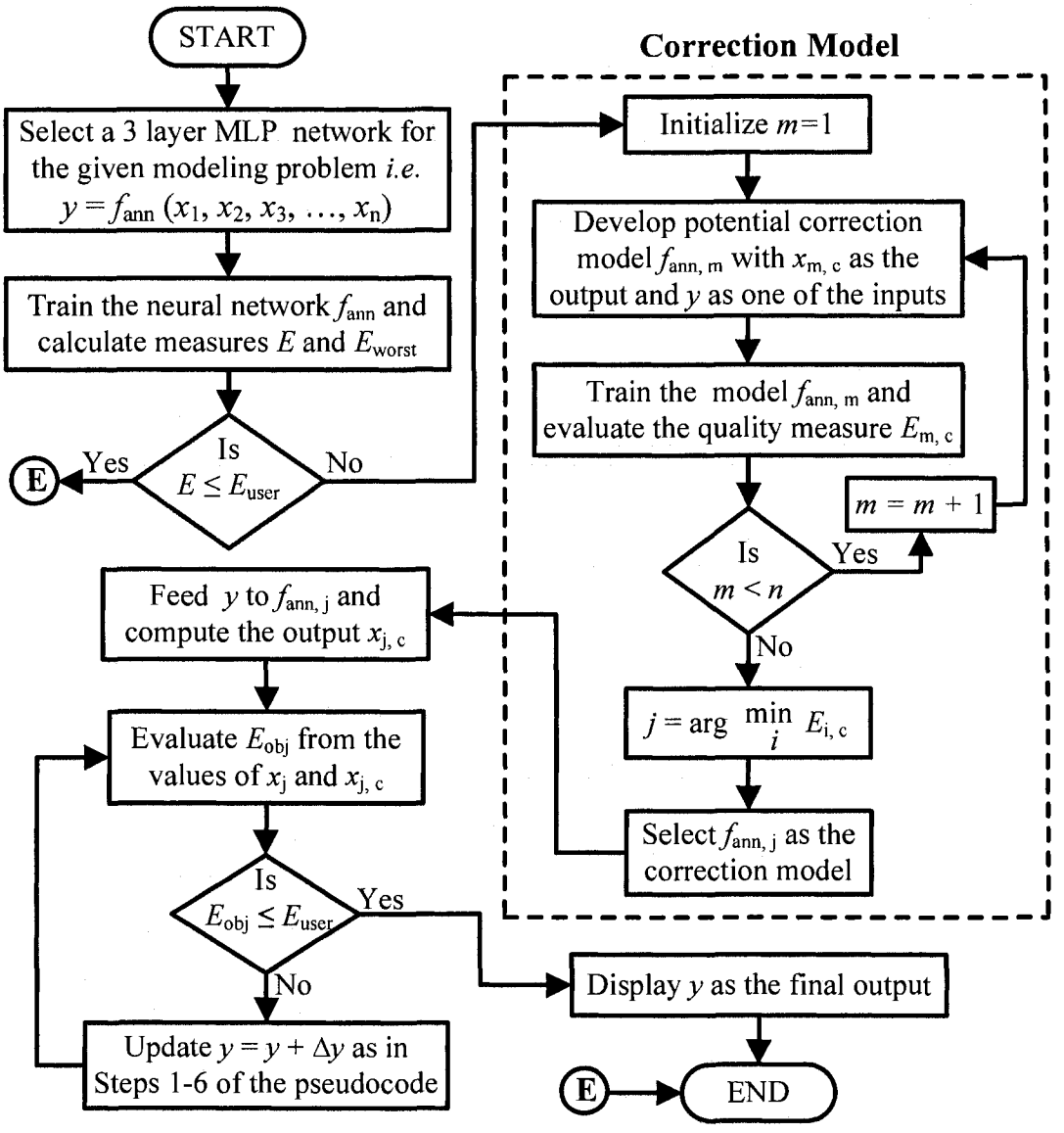


Figure 4.2 Flow-chart illustrating the proposed modeling approach.

PseudoCode:

Step 1: Given a new data $(x_1, x_2, ..., x_n)$, f_{ann} is used to determine an approximate output y. Initialize $\Delta y = \alpha > 0$ and $h = 1$.

Step 2: Data inputs $(x_1, x_2, ..., x_{j-1}, y_1, x_{j+1}, ..., x_n)$ are fed to the correction model $f_{ann,\,j}$ leading to an output $x_{j,\,c}$.

Step 3: Evaluate the objective function $E_{obj} = \dfrac{x_j - x_{j,\,c}}{|x_j|} \times 100$. If $E_{obj} \le E_{user}$, RETURN y (*i.e.* neural model output).

Step 4: Set $y_1 = y - \Delta y$, $y_2 = y + \Delta y$, $h_{new} = \dfrac{E_{obj}}{|E_{obj}|}$,

$a = f_{ann,\,j}(x_1, ..., x_{j-1}, y_1, x_{j+1}, ..., x_n)$, $b = f_{ann,\,j}(x_1, ..., x_{j-1}, y_2, x_{j+1}, ..., x_n)$,

and $\delta = \dfrac{b-a}{|b-a|}$.

Step 5: If $h \neq h_{new}$, set $h = h_{new}$ and $\alpha = \alpha/2$.

Step 6: Set $\Delta y = \alpha * h * \delta$ and $y = y + \Delta y$. GOTO *Step 2*.

4.4 ILLUSTRATION EXAMPLE (MOSFET)

Layout of MOSFETs is critical at high-frequencies, at which, extrinsic/intrinsic elements become functions of geometry [36]. A compact ANN model of a MOSFET is to be developed *i.e.*

$$W/L = f_{ann}(L, V_{DS}, I_D), \tag{4.7}$$

where L and W represent gate length and width, V_{DS} represents drain voltage, and I_D represents drain current. Gate voltage V_{GS} is kept fixed at 0.9V. Employing a detailed model in *HSPICE*, training data is generated. In other words, a datasheet of I_D is generated by varying W/L, L, and V_{DS} along uniform grids. This datasheet can be rearranged (as required) during the training of desired as well as potential correction neural models.

First, a 3-layer MLP with three inputs and one output (see Fig. 4.3(a)) is trained using *NeuroModeler*. For example, training data when L = 0.5μm is shown in Fig. 4.3(b) and the corresponding f_{ann} responses are shown in Fig. 4.3(c). Assuming E_{user} = 5%, the stand-alone f_{ann} exhibits unacceptable E and E_{worst} (>10% and >38% respectively). Out of the three potential correction models developed for this modeling scenario,

$$I_D = f_{ann,3}(L, V_{DS}, W/L) \tag{4.8}$$

is chosen as the correction model according to (4.6) and Table 4.1. Finally, f_{ann} is used in conjunction with $f_{ann,3}$ thereby resulting in an acceptable model. Initial value of Δy is set to be 1 during model utilization, and iteratively updated as in the pseudocode. Improved responses when L = 0.5μm are shown in Fig. 4.3(d). As seen in Table I, E and E_{worst} based on the proposed approach are significantly improved (*i.e.* < 0.5% and < 2% respectively).

TABLE 4.1
COMPARISON OF MODEL ACCURACIES FOR THE MOSFET EXAMPLE

Error	Stand-Alone Desired Model	Potential Correction Models			Proposed Approach
		L	V_{DS}	I_D	
E	10.08%	2.84%	2.56%	0.45%	0.41%
E_{worst}	38.7%	69.21%	50.14%	1.89%	1.98%

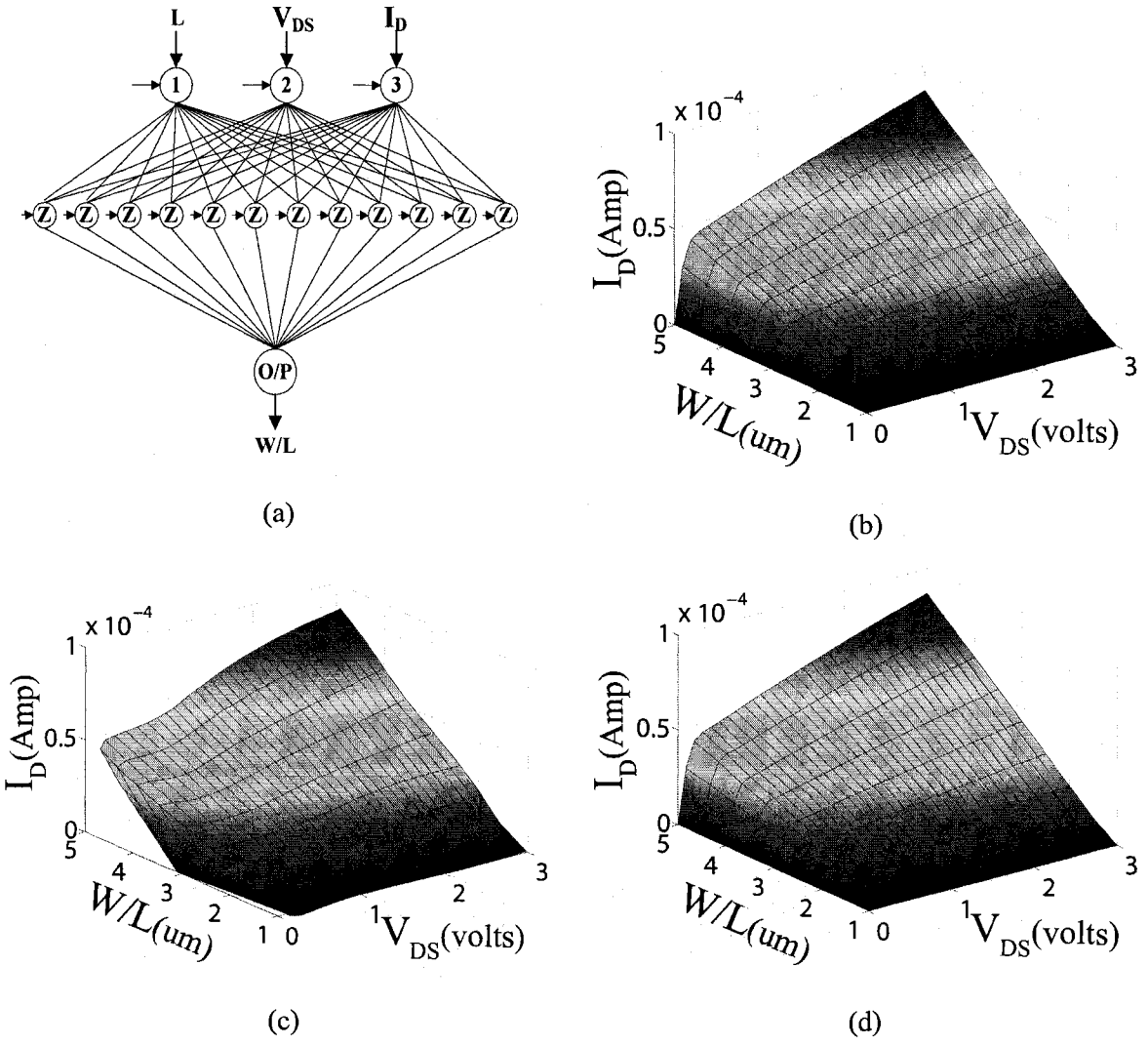


Figure 4.3 (a) A 3-layer MLP representing the structure of the desired MOSFET model, (b) Training data corresponding to $L = 0.5\mu m$, (c) Responses of the stand-alone neural model, and (d) Responses of the proposed neural model.

4.5 SUMMARY

In this chapter, a new and systematic ANN modeling approach based on a novel correction model concept has been proposed. The format/structure of the correction model is determined in a logical fashion, and the correction model is employed together with the stand-alone desired model for improving the accuracy of the desired model.

Simple 3-layer MLP networks are used for the development of desired as well as correction models. As such, computational time in the training of potential correction models can be neglected. The proposed approach eliminates the need for in-depth understanding of advanced ANN concepts. Examples confirm that the proposed approach leads to accurate neural models while keeping the ANN model structure simple.

CHAPTER 5

DESIGN TOOL AND EXAMPLES

5.1 INTRODUCTION

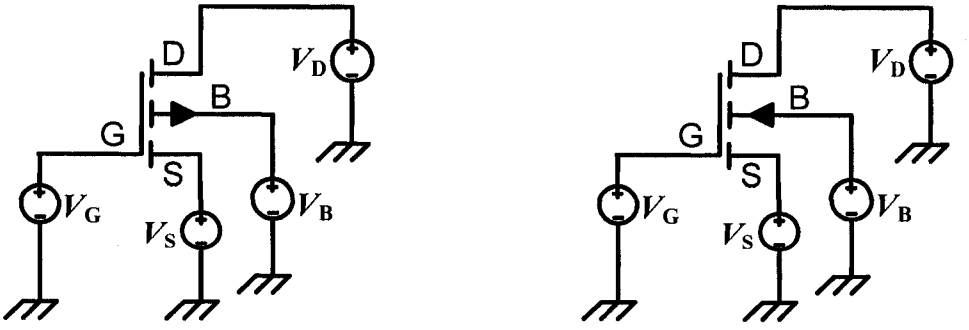
Proposed modeling algorithms (see chapters 3 and 4) are employed for developing accurate and compact MOSFET models from physics based device/circuit simulator data. The prime objectives of this chapter can best be understood by referring back to Fig.1.3 in chapter 1 of this thesis. Beginning with the technological information (*i.e.* 0.5μm or 0.18μm CMOS), training data for MOSFET modeling is obtained from a well known circuit simulator *i.e. HSPICE*. Using such data, accurate neural models are developed employing proposed algorithms (chapter 3 and 4). These neural models are incorporated in the development of analog IC basic building block modules (*e.g.* current sinks/sources,

simple and cascode current mirrors, single stage amplifiers, voltage dividers, differential amplifiers, and three stage operational amplifiers) leading to the design tool emerging from this thesis. Various development phases of the design tool, starting with the training data generation, efficient processing of the training data, and modeling of MOSFETs are briefly discussed below. Applications of the tool for deriving corresponding geometrical dimensions of several MOSFET IC building blocks, given some specifications, in CMOS technologies (0.5μm and 0.18μm) are presented.

5.2 Design Phases of the Tool

5.2.1 Simulation

CMOS transistors, namely N-Channel MOS (NMOS) and P-Channel MOS (PMOS) (see Fig. 5.1) are simulated in Cadence's well known simulator (*i.e. HSPICE*) for 0.5μm and 0.18μm technologies and a detailed datasheet of drain current I_D is generated by sweeping the transistor dimensions and terminal voltages (*i.e.* L, W/L, V_D, V_G, and V_B). V_{GS} and V_{DS} are varied from 0 to 1.5 volts in the steps of 0.1 volts. Detailed description about the parameters swept and their ranges is provided in Table 5.1.



(a) (b)

Figure 5.1 Transistors (a) PMOS and (b) NMOS simulated in the Cadence's HSpice simulator.

Table 5.1 PARAMETERS AND RANGES USED FOR THE DESIRED MODEL

Parameter	Range	Step
Effective Length (L)	0.5 micron to 2 micron	0.1 micron
W/L Ratio	1 to 500	1
Bulk-Source voltage (V_{BS}/ V_{SB})	0 Volts to -1.5 Volts	-0.1 Volts
Gate-Source Voltage (V_{GS}/V_{SG})	0 Volts to 1.5 Volts	0.1 Volts
Drain-Source Voltage (V_{DS}/V_{SD})	0 Volts to 1.5 Volts	0.1 Volts

5.2.2 Efficient Processing

There were a few challenges in the creation of the training data set, for instance too many parameters to sweep, a plethora of data to be handled, and tedious task of database generation from the *HSPICE* output files. In order to resolve these problems, a script is written in UNIX operating system to automatically sweep all the parameters and store the output files systematically. A software program is then developed in Microsoft Visual C# .Net programming language (see Fig. 5.2). The program requests the folder containing all the simulation output files (.lis files), generated by the script from *HSPICE,* as the input. The program recursively opens each of the simulation output files, parses the data and stores it in the database. As a result, training database, to be used for development of the neural models, is generated.

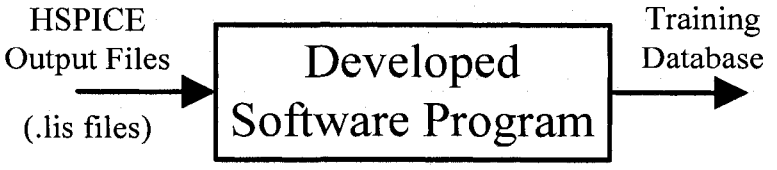


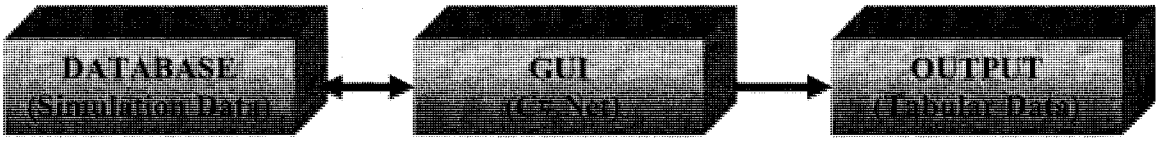
Figure 5.2 Block diagram of the developed software program.

5.2.3 Neural Modeling

Neural networks are employed for modeling the database generated from the afore-mentioned steps. *NeuroModeler* is used for the development of neural models. Conjugate gradient training algorithm from the *NeuroModeler* is used for training the neural models. Simple MLP structure failed to generate satisfactory neural models for the database, and therefore new modeling algorithms based on binning concepts (see Chapter 3), and a new neural modeling approach based on correction model concept (see Chapter 4) have been employed for modeling the database. Proposed modeling algorithms help to satisfactorily model the database with simple 3-layer MLP neural networks.

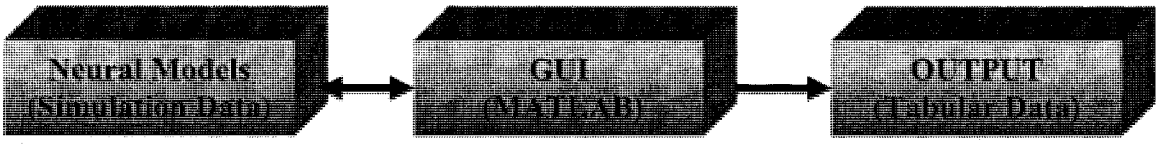
5.2.4 Programming Interface

In this phase, all the developed neural models are integrated to develop a design tool to aid the analog designers. Graphical User Interfaces (GUIs) are also designed, for the tools developed for both 0.5μm and 0.18μm technologies, in two distinct programming environments namely, MATLAB and C#.Net. For the users with a MATLAB license, the design tool is created in MATLAB environment with the extensive support of MATLAB libraries and graphics. For the users not having a MATLAB license, the design tool is also developed in C#.Net that can be used as a standalone tool. Several analog IC basic building block modules are developed for the tool. However, the design tools are still in primary stages of development and enhancements of the design tools are discussed as the future work/extension to this thesis. Block diagram representation of the tools developed using C#.Net and MATLAB are depicted in Fig 5.3(a) and Fig. 5.3(b) respectively. Design examples of various analog IC building blocks for both 0.5μm and 0.18μm CMOS technologies are presented in the following sections of this chapter.



Back-End **Front-End** **Display**

(a)



Back-End **Front-End** **Display**

(b)

Figure 5.3 Architectures of the tools designed using (a) C#.Net and (b) MATLAB interfaces are depicted.

5.3 Design Examples for 0.5 Micron Technology

Illustration examples showing the design of various analog IC building blocks (*e.g.* current sinks/sources, current mirrors, single stage amplifiers, *etc.*) for 0.5μm technology using the design tool are shown below. Two neural models, namely, f_{ann} and $f_{ann,\,5}$ are developed and used according to the modeling approach proposed in chapter 4. If the user provides any preference for L, the corresponding value of W/L is provided as the output, otherwise values of W/L for various values of L are provided as the output to the user.

$$W/L = f_{ann}(L, V_{SB/BS}, V_{SG/GS}, V_{SD/DS}, I_{SD/DS}), \tag{5.1}$$

and

$$I_{SD/DS} = f_{ann}(L, V_{SB/BS}, V_{SG/GS}, V_{SD/DS}, W/L). \tag{5.2}$$

5.3.1 Current Sources

Current sources have a wide range of applications in the analog circuits, for instance single-stage amplifiers, differential amplifiers, *etc.* A MOS transistor (NMOS or PMOS) operating in saturation can act as a current source. To design a stable current source, a DC bias voltage is usually applied to the gate of the MOS transistor (see Fig. 5.4).

A typical problem statement for the design of a basic current source can be stated as

$$[L, W/L] = f(V_{SB/BS}, V_{SG/GS}, V_{SD/DS}, I_{SD/DS}). \tag{5.3}$$

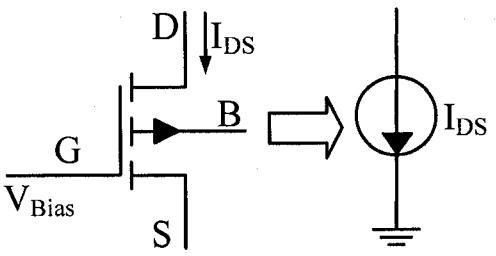


Figure 5.4 Basic Current Source and its equivalent representation.

A Current source is realized using both PMOS as well as NMOS separately in this example. User specifications are V_S = 0 V, V_B = 0 V, $V_{SG/GS}$ = 1.1 V, $V_{SD/DS}$ = 1.0 V, and $I_{SD/DS}$ = 25 μA. One of the result combinations obtained from the tool for both PMOS and NMOS along with the corresponding simulation verification results from *HSPICE* are tabulated (see Table 5.2). With the same biasing conditions, it can be noted that a PMOS requires larger area than an NMOS.

Table 5.2 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR CURRENT SOURCE

Device Type	**$V_{SB/BS}$ (Volts)**	**$V_{SG/GS}$ (Volts)**	**$V_{SD/DS}$ (Volts)**	**L (μm)**	**W (μm)**	**$I_{SD/DS}$ (Desired) (Amp)**	**$I_{SD/DS}$ (Modeled) (Amp)**	**$I_{SD/DS}$ (Simulated) (Amp)**
PMOS	0	1.1	1.0	0.9	18.9	25E-6	25.1E-06	25.1E-06
NMOS	0	1.1	1.0	0.9	0.9	25E-6	25.1E-06	25.1E-06

5.3.2 Simple Current Mirrors

Current mirrors find a wide variety of applications in current-output based active devices, *e.g.* operational transconductance amplifiers (OTAs), as the output stage [37]. Simple current mirrors are critical components as both linearity and output resistance of the output stage depend on their performance [38]. A simple NMOS current mirror and the circuit simulated in *HSPICE* are shown in Fig. 5.5(a) and Fig. 5.5(b) respectively.

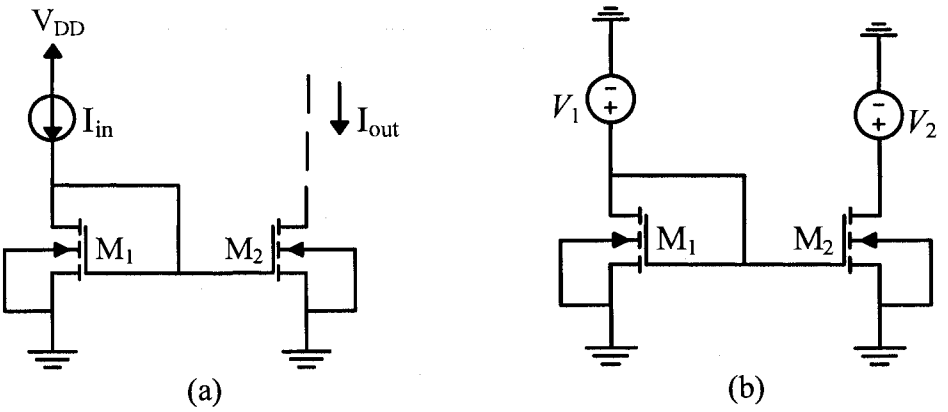


Figure 5.5 (a) Simple NMOS Current Mirror, and (b) circuit simulated in *HSPICE*.

Typical specifications for a current mirror might include (i) g_{ds} for a V_{DSmin}, (ii) max area, (iii) deterministic I_{in}-I_{out} mistrack, (iv) for max g_m for noise considerations, (v) max g_m ΔV_t for random mismatch, *etc.* In this work, transistor sizes are designed for a given voltage and current specification only. However, the aforementioned specifications can be considered for the extensive design of current mirrors as an extension to this work. A typical problem statement for the design of a simple current mirror can be stated as

$$[L, W/L] = f(V_{GS_{M1}}, V_{DS_{M1}}, V_{BS}, I_{in}, I_{out}). \tag{5.4}$$

Values of W/L for various values of L are calculated for both M_1 and M_2 using f_{ann} and $f_{ann,\,5}$ (see (5.1) and (5.2)), and those satisfying the functional relation between I_{in} and I_{out} are provided as the outputs.

In this example, an NMOS current mirror is designed for the given user specifications of $V_{GS_{M1}} = V_{DS_{M1}} = V_{GS_{M2}} = 0.7$ V, $V_{S_{M1}} = V_{S_{M2}} = 0$ V, $V_{BS_{M1}} = V_{BS_{M2}} = 0$ V, $I_{in} = 5$ µA, and $I_{out} = 2 \times I_{in}$. One of the result combinations obtained from the tool and the corresponding simulation verifications from *HSPICE* are tabulated (see Table 5.3).

Table 5.3 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR SIMPLE NMOS CURRENT MIRROR

Device	V_{BS} (Volts)	V_{GS} (Volts)	V_{DS} (Volts)	L (µm)	W (µm)	$I_{in/out}$ (Desired) (Amp)	$I_{in/out}$ (Modeled) (Amp)	$I_{in/out}$ (Simulated) (Amp)
M_1	0	0.7	0.7	2.0	30.0	5 E-6	5E-6	5E-6
M_2	0	0.7	1.0	2.0	60.0	10 E-6	10E-5	10.3E-6

5.3.3 Cascode Current Mirror

Cascode current mirrors offer increased output impedance as compared to the simple current mirrors [39]. An NMOS cascode current mirror and the circuit simulated using *HSPICE* are depicted in Fig. 5.6(a) and Fig. 5.6(b) respectively.

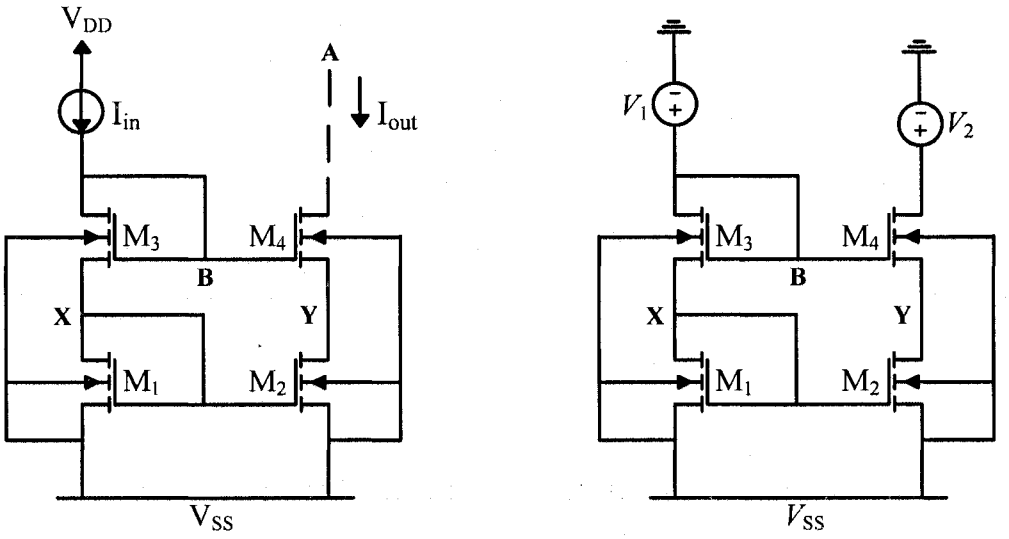


Figure 5.6 (a) Cascode Current Mirror, and (b) circuit simulated in *HSPICE*.

A typical problem statement for the design of a cascode current mirror is stated as

$$[L, W/L] = f(V_X, V_Y, V_{GS_{M3}}, I_{in}, I_{out}). \quad (5.5)$$

Values of W/L for various values of L are calculated for M_1, M_2, M_3 and M_4 using f_{ann} and $f_{ann,\,5}$ (see (5.1) and (5.2)), and that satisfying the I_{in}/I_{out} ratio are provided as outputs.

An NMOS cascode current mirror is designed in this example, for the following user specifications $V_X = V_Y = 0.7$ V, $V_{GS_{M3}} = V_{DS_{M3}} = V_{GS_{M4}} = 0.7$ V, $V_{SS} = 0$ V, $I_{in} = 10$ μA, and $I_{out} = I_{in}$. One of the result combinations obtained from the tool and the corresponding simulation verification results from *HSPICE* are tabulated (see Table 5.4).

Table 5.4 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR CASCODE CURRENT MIRROR

Device	V_{BS} (Volts)	V_{GS} (Volts)	V_{DS} (Volts)	L (μm)	W (μm)	$I_{in/out}$ (Desired) (Amp)	$I_{in/out}$ (Modeled) (Amp)	$I_{in/out}$ (Simulated) (Amp)
M_1, M_2,	0	0.7	0.7	1	14	10E-6	10E-6	10.1E-6
M_3, M_4	-0.7	0.7	0.7	0.5	7	10E-6	10E-6	10.1E-6

5.3.4 Common Drain Amplifier

Common drain amplifiers, used as voltage buffers or source followers, ideally have a small signal voltage gain close to unity [40]. Input and output of this amplifier is located at gate and source terminals respectively hence it is commonly known as common drain amplifier. A common drain amplifier with a current mirror active load, its equivalent small signal model and the circuit simulated using *HSPICE* are depicted in Fig. 5.7(a), Fig. 5.7(b), and Fig. 5.7(c) respectively. Gain A_v of the amplifier is computed using

$$A_v = \frac{v_{out}}{v_{in}} = \frac{g_{m1}}{g_{m1} + g_{mb1} + g_{ds1} + g_{ds2}}. \quad (5.6)$$

A typical problem statement for the design of a common drain amplifier is stated as

$$[L,\ W/L] = \boldsymbol{f}(V_{DD},\ V_{BS_{M1}},\ V_{BS_{M3}},\ V_{GS_{M3}},\ A_V,\ I_{Bias}). \tag{5.7}$$

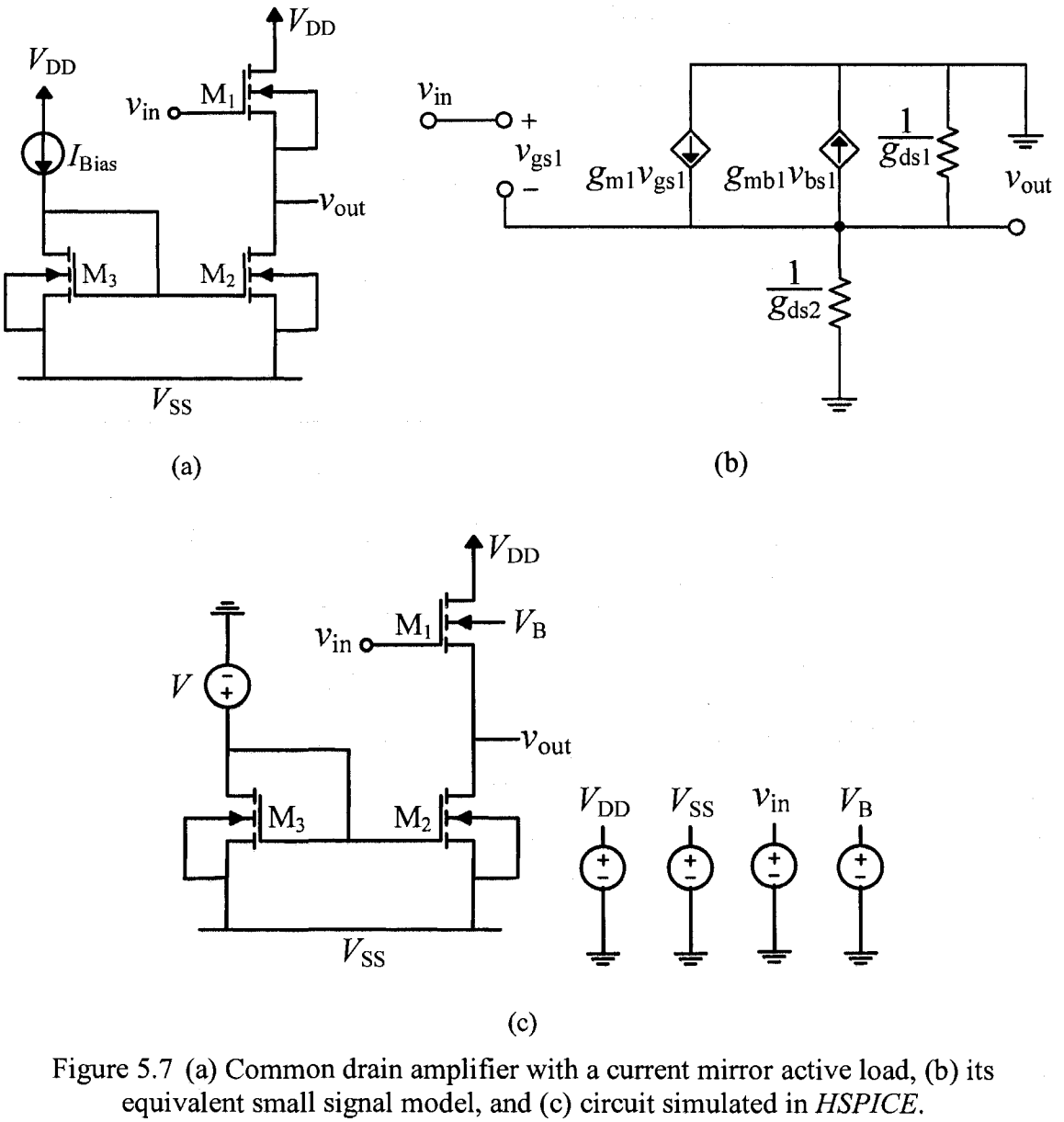


Figure 5.7 (a) Common drain amplifier with a current mirror active load, (b) its equivalent small signal model, and (c) circuit simulated in *HSPICE*.

Values of W/L for various values of L are calculated for M_1, M_2, and M_3 using f_{ann} and $f_{ann,\,5}$ (see (5.1) and (5.2)), are fed to $\boldsymbol{f_{ann2}}$ to calculate the gain parameters as

$$[g_m,\ g_{ds},\ g_{mb}] = \boldsymbol{f_{ann2}}(L,\ W/L,\ V_{SB/BS},\ V_{SG/GS},\ V_{SD/DS},\ I_{SD/DS}). \tag{5.8}$$

Only those combination values of drain voltages ($V_{DS_{M1}}$ and $V_{DS_{M2}}$), that satisfy the KVL (*i.e.* $V_{DS_{M1}} + V_{DS_{M2}} = V_{DD}$) are considered. Finally, the combinations of values of L and W/L satisfying the required gain criterion are provided as the outputs.

A common drain amplifier is designed in this example, for the following user specifications $V_{DD} = 1.5$ V, $V_{SS} = 0$ V, $V_{BS_{M1}} = V_{BS_{M2}} = V_{BS_{M3}} = 0$ V, $V_{GS_{M3}} = V_{DS_{M3}} = V_{GS_{M2}} = 0.8$ V, $A_V \approx 1$, and $I_{Bias} \leq 15$ μA. For this design, a double well 0.5 μm process is required. One of the combinations obtained from the tool and the corresponding simulation verifications from *HSPICE* are tabulated (see Table 5.5).

Table 5.5 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR COMMON DRAIN AMPLIFIER WITH ACTIVE LOAD

Device	V_{BS} **(Volts)**	V_{GS} **(Volts)**	V_{DS} **(Volts)**	L **(μm)**	W **(μm)**	I_{Bias} **(Simulated) (Amp)**	A_v **(Modeled) (V/V)**	A_v **(Simulated) (V/V)**
M_1	0	0.7	0.8	0.7	4.9	11.2E-6	0.9617	0.962
M_2	0	0.8	0.7	1.9	9.5	11.3E-6 (Modeled)		
M_3	0	0.8	0.8	1.9	9.5			

5.3.5 Common Source Amplifier

Common source amplifier is the most popular gain stage, especially when high input impedance is one of the design requirements [40]. Active load helps to realize the high impedance output load without excessively large resistors or a large supply voltage. A common source amplifier with a current mirror active load, its equivalent small signal model and the circuit simulated using *HSPICE* are depicted in Fig. 5.8(a), Fig. 5.8(b), and Fig. 5.8(c) respectively. Gain of common source amplifier A_v is computed using

$$A_v = \frac{v_{out}}{v_{in}} = -\frac{g_{m1}}{g_{ds1} + g_{ds2}} \tag{5.9}$$

A typical problem statement for the design of a common source amplifier is stated as

$$[L, W/L] = f(V_{DD}, V_{BS_{M1}}, V_{SB_{M2}}, V_{SB_{M3}}, V_{SD_{M3}}, A_V, I_{Bias}) \tag{5.10}$$

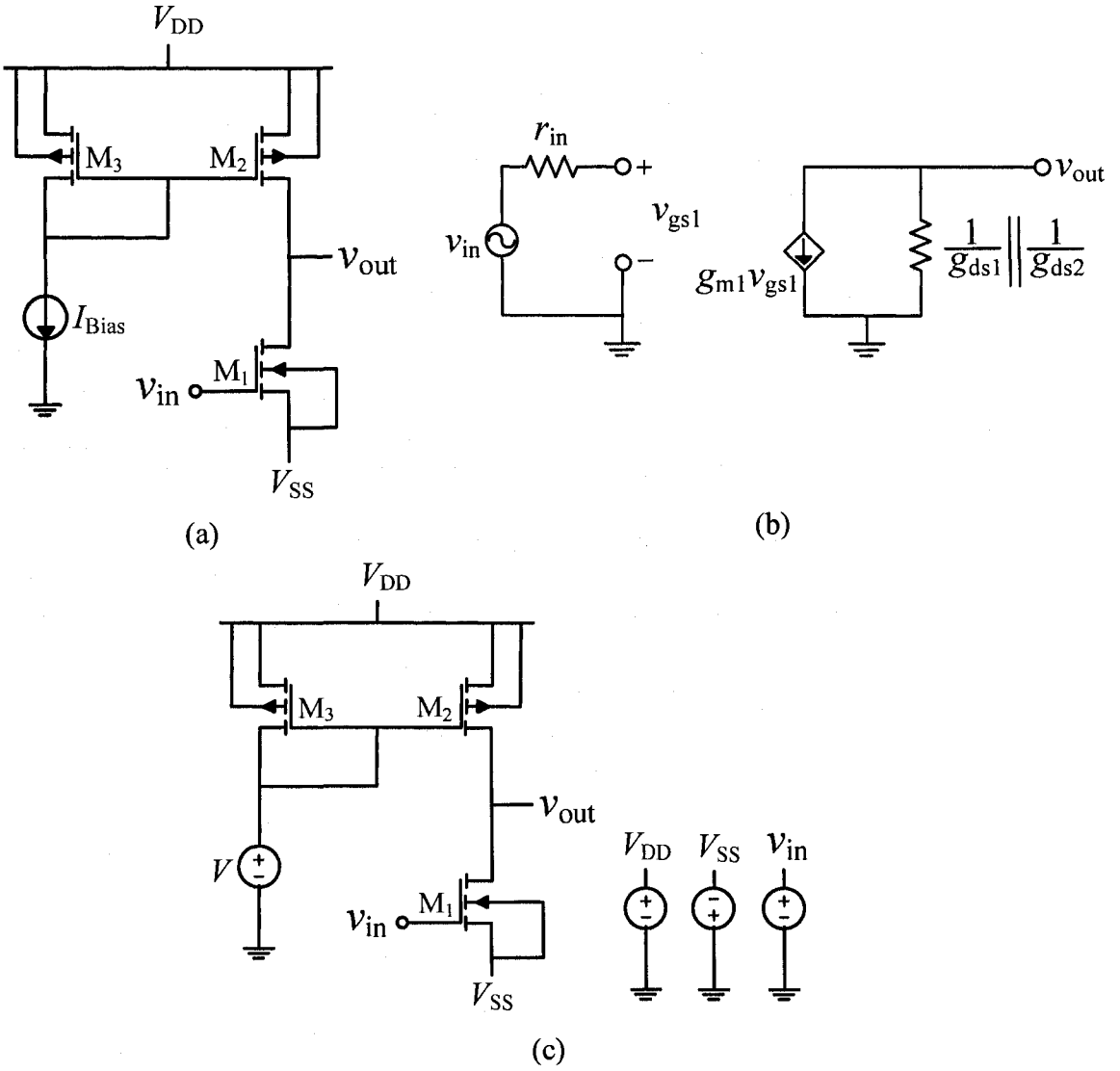


Figure 5.8 (a) Common source amplifier with a current mirror active load, (b) its equivalent small signal model, and (c) circuit simulated in *HSPICE*.

Values of W/L for various values of L are computed for M_1 and M_2 using f_{ann} and $f_{ann,\,5}$ (see (5.1) and (5.2)), are fed to $\boldsymbol{f_{ann2}}$ (see (5.8)) and gain parameters are calculated.

In this example, a common source amplifier is designed for the user specifications of $V_{DD} = 1.5$ V, $V_{SS} = 0$ V, $V_{BS_{M1}} = V_{SB_{M2}} = V_{SB_{M3}} = 0$ V, $V_{SG_{M3}} = V_{SD_{M3}} = V_{SG_{M2}} = 1.1$ V, $|A_V| \geq 25$,

and $I_{Bias} \leq 15$ μA. Similar to the common drain amplifier, KVL condition is satisfied for the selection of design parameters. One of the combinations obtained from the tool and the corresponding simulation verifications from *HSPICE* are tabulated (see Table 5.6).

Table 5.6 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR COMMON SOURCE AMPLIFIER WITH ACTIVE LOAD

Device	$V_{SB/BS}$ (Volts)	$V_{SG/GS}$ (Volts)	$V_{SD/DS}$ (Volts)	L (μm)	W (μm)	I_{Bias} (Simulated) (Amp)	A_v (Modeled) (V/V)	A_v (Simulated) (V/V)
$M_{1(NMOS)}$	0	0.7	0.7	0.7	7.0	14.9E-6		
$M_{2(PMOS)}$	0	1.1	0.8	1.7	28.9	14.9E-6 (Modeled)	−25.55	−25.83
$M_{3(PMOS)}$	0	1.1	1.1	1.7	28.9			

5.3.6 Common Gate Amplifier

A common gate amplifier is most commonly used gain stage, when relatively small input impedance is desired [40]. A common gate amplifier with a current mirror active load, its equivalent small signal model and the circuit simulated in *HSPICE* are depicted in Fig. 5.9(a), Fig. 5.9(b), and Fig. 5.9(c) respectively. Gain A_v is of the amplifier can be computed using

$$A_v = \frac{v_{out}}{v_{in}} = \frac{g_{m1} + g_{mb1} + g_{ds1}}{g_{ds1} + g_{ds2}}. \tag{5.11}$$

A typical problem statement for the design of a common gate amplifier is stated as

$$[L, W/L] = f(V_{DD}, V_{BS_{M1}}, V_{SB_{M2}}, V_{SB_{M3}}, V_{SD_{M3}}, A_V, I_{Bias}). \tag{5.12}$$

Values of W/L for various values of L are computed for M_1 and M_2 using f_{ann} and $f_{ann,\,5}$ (see (5.1) and (5.2)) are fed to f_{ann2} (see (5.8)) and gain parameters are calculated. The values of L and W/L satisfying the gain criterion are provided as the outputs.

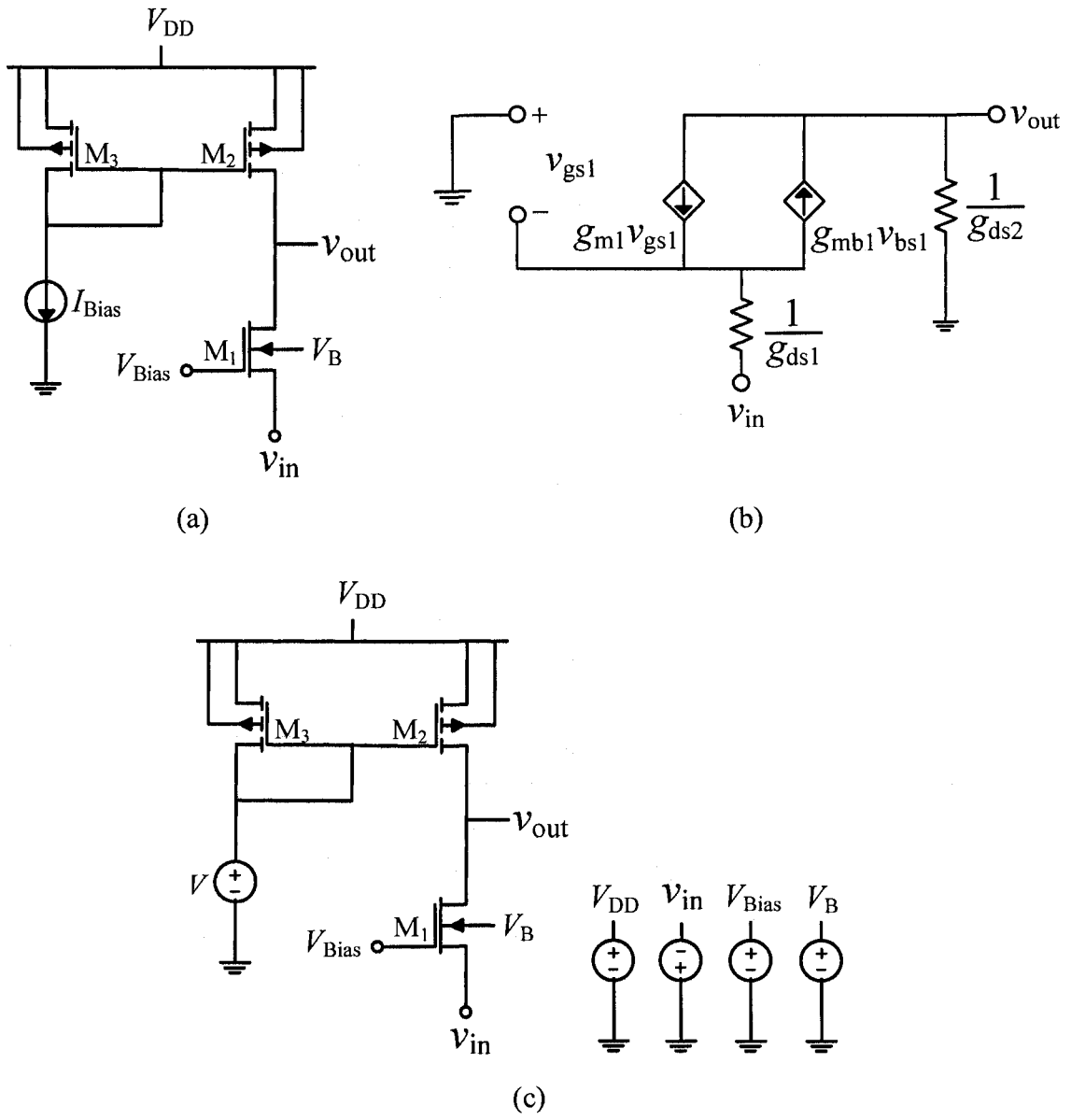


Figure 5.9 (a) Common gate amplifier with a current mirror active load, (b) its equivalent small signal model, and (c) circuit simulated in *HSPICE*.

A common gate amplifier is designed in this example for the user specifications of $V_{DD} = -V_{SS} = 1.2$ V, $V_{BS_{M1}} = V_{SB_{M2}} = V_{SB_{M3}} = 0$ V, $V_{SG_{M3}} = V_{SD_{M3}} = V_{SG_{M2}} = 1.1$ V, $A_V \geq 60$, and $I_{Bias} \leq 5$ μA. KVL is satisfied similar to the previous designs. A twin well 0.5 μm CMOS process is needed for the design. One of the combinations obtained from the tool and the corresponding simulation verifications from *HSPICE* are tabulated (see Table 5.7).

Table 5.7 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR COMMON GATE AMPLIFIER WITH ACTIVE LOAD

Device	V_{BS} (Volts)	$V_{GS/SG}$ (Volts)	$V_{DS/SD}$ (Volts)	L (μm)	W (μm)	I_{Bias} (Simulated) (Amp)	A_v (Modeled) (V/V)	A_v (Simulated) (V/V)
$M_{1(NMOS)}$	0	0.7	1.1	0.9	0.9	1E-6	60.408	60.952
$M_{2(PMOS)}$	0	1.1	1.3	1.7	1.7	1E-6 (Modeled)		
$M_{3(PMOS)}$	0	1.1	1.1	1.7	1.7			

5.3.7 Push-Pull Amplifier

A push-pull amplifier implements two complementary transistors connected together as shown in Fig. 5.10(a). A push-pull amplifier is commonly used in applications where high power output and good fidelity are critical in design specifications [41], for instance, receiver output stages, AM modulators, *etc.* An equivalent small signal model of a push-pull amplifier is shown in Fig. 5.10(b). Gain A_v of the push-pull amplifier is computed as

$$A_v = \frac{v_{out}}{v_{in}} = -\frac{g_{m1} + g_{m2}}{g_{ds1} + g_{ds2}}. \tag{5.13}$$

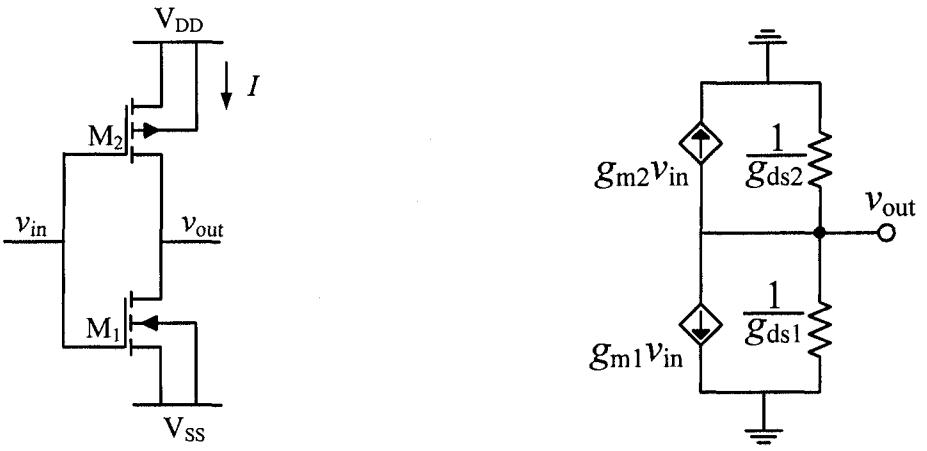
A typical problem statement for the design of a push-pull amplifier can be stated as

$$[L, W/L] = f(V_{DD}, A_v, I). \tag{5.14}$$

Values of W/L for various values of L are calculated for M_1 and M_2 using f_{ann} and $f_{ann,5}$ (see (5.1) and (5.2)) are fed to f_{ann2} (see (5.8)) and gain parameters are calculated. Values of L and W/L satisfying the gain criterion are provided as outputs.

In this example, a push-pull amplifier is designed for the following given user specifications, $V_{DD} = -V_{SS} = 1.2$ V, $A_v \geq 375$ V/V, and $I \leq 100$ μA. One of the possible result combinations from the tool and the corresponding simulation verifications are tabulated (see Table 5.8). Threshold voltage (V_{th}) for submicron transistors appears to

depend not only on V_{BS}, but also on other dc voltages when connected in a totem pole configuration. This may force some transistors of the circuit to operate in regions other than saturation. Hence inconsistency with *HSPICE* results is possible in some cases.



(a) (b)

Figure 5.10 (a) A push-pull amplifier, and (b) its equivalent small signal model

Table 5.8 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR PUSH-PULL AMPLIFIER

Device	V_{BS} **(Volts)**	$V_{GS/SG}$ **(Volts)**	$V_{DS/SD}$ **(Volts)**	L **(μm)**	W **(μm)**	I_{Bias} **(Modeled) (Amp)**	I_{Bias} **(Simulated) (Amp)**	A_v **(Simulated) (V/V)**
$M_{1(NMOS)}$	0	1.2	1.4	1.7	5.1	72.6E-6	73.6E-6	– 384.96
$M_{2(PMOS)}$	0	1.2	1.0	1.4	50.4			

5.3.8 Validations for 0.5μm Technology

Current sources/sinks have been tested for numerous values in between 0.17 nA and 0.578 A for NMOS, and between 1.37 pA and 0.158 mA for PMOS. Current mirrors have shown good agreement for currents from 1 μA to 100 μA. Amplifier circuits have also been tested successfully for numerous gain values ranging from, 0.9 for the common drain amplifier to 400 for the push-pull amplifier.

5.4 DESIGN EXAMPLES FOR 0.18 MICRON TECHNOLOGY

Illustration examples showing the design of various analog IC building blocks (*e.g.* current sinks/sources, current mirrors, single stage amplifiers, voltage divider, differential amplifier, *etc.*) for 0.18μm technology using the design tool are presented. The circuit schematics are the same as provided for 0.5μm technology examples.

5.4.1 Current Sources

A typical problem statement for the design of a basic current source can be stated as

$$[L, W/L] = f(V_{SB/BS}, V_{SG/GS}, I_{SD/DS}). \tag{5.15}$$

Current source is realized using both PMOS and NMOS in this example for the user specifications, $V_{S_{NMOS}} = -1.2$ V, $V_{S_{PMOS}} = 1.2$ V, $V_{BS_{NMOS}} = V_{SB_{PMOS}} = 0$ V, $V_{SG/GS} = 1.0$ V, $V_{SD/DS} = 0.7$ V, and $I_{SD/DS} = 20$ μA. One of the combinations for both PMOS and NMOS obtained from the tool and the corresponding simulation verifications from *HSPICE* for both the cases are tabulated (see Table 5.9).

Table 5.9 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR CURRENT SOURCE

Device Type	$V_{SB/BS}$ (Volts)	$V_{SG/GS}$ (Volts)	$V_{SD/DS}$ (Volts)	L (μm)	W (μm)	$I_{SD/DS}$ (Desired) (Amp)	$I_{SD/DS}$ (Modeled) (Amp)	$I_{SD/DS}$ (Simulated) (Amp)
PMOS	0	1.0	0.7	0.7	27.3	20E-6	20.2E-06	20.2E-06
NMOS	0	1.0	0.7	0.7	0.7	20E-6	20.2E-06	20.2E-06

5.4.2 Simple Current Mirrors

A typical problem statement for the design of a basic current source can be stated as

$$[L, W/L] = f(V_{GS_{M1}}, V_{DS_{M1}}, V_{BS}, I_{in}, I_{out}). \tag{5.16}$$

In this example, an NMOS current mirror is designed for the given user specifications of $V_{GS_{M1}} = V_{DS_{M1}} = V_{GS_{M2}} = 1.1$ V, $V_{S_{M1}} = V_{S_{M2}} = 0$ V, $V_{BS_{M1}} = V_{BS_{M2}} = -1.2$ V, $I_{in} = 20$ μA, and $I_{out} = 3 \times I_{in}$. One of the result combinations obtained from the tool and the corresponding simulation verifications from *HSPICE* are tabulated (see Table 5.10).

Table 5.10 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR NMOS CURRENT MIRROR

Device	V_{BS} (Volts)	V_{GS} (Volts)	V_{DS} (Volts)	L (μm)	W (μm)	$I_{in/out}$ (Desired) (Amp)	$I_{in/out}$ (Modeled) (Amp)	$I_{in/out}$ (Simulated) (Amp)
M_1	-1.2	1.1	1.1	1.1	5.5	20 E-6	19.7E-6	19.7E-6
M_2	-1.2	1.1	1.0	1.1	16.5	60 E-6	59.1E-6	59.8E-6

5.4.3 Cascode Current Mirror

A typical problem statement for the design of a basic current source can be stated as

$$[L, W/L] = f(V_X, V_Y, V_{GS_{M3}}, V_{DS_{M3}}, I_{in}, I_{out}). \quad (5.17)$$

In this example, an NMOS cascode current mirror is designed for the following user specifications, $V_X = V_Y = 0.6$ V, $V_{BS_{M1}} = V_{BS_{M2}} = 0$ V, $V_{BS_{M3}} = V_{BS_{M4}} = -0.6$ V, $V_{SS} = 0$ V, $V_{GS_{M3}} = V_{DS_{M3}} = V_{GS_{M4}} = 0.7$ V, $I_{in} = 100$ μA, and $I_{out} = I_{in}$. One of the result combinations obtained from the tool and the corresponding simulation verifications from *HSPICE* are tabulated (see Table 5.11).

Table 5.11 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR CASCODE CURRENT MIRROR

Device	V_{BS} (Volts)	V_{GS} (Volts)	V_{DS} (Volts)	L (μm)	W (μm)	$I_{in/out}$ (Desired) (Amp)	$I_{in/out}$ (Modeled) (Amp)	$I_{in/out}$ (Simulated) (Amp)
M_1, M_2,	0	0.6	0.6	0.3	0.6	10E-5	10.1E-5	10.1E-5
M_3, M_4	-0.6	0.7	0.7	0.4	6.0	10E-5	10.1E-5	10.1E-5

5.4.4 Voltage Divider

Voltage dividers have primary application in the DC biasing of various circuits. A voltage divider designed depicted in Fig. 5.11 is designed in this example.

A typical problem statement for the design of a voltage divider can be stated as

$$[L, W/L, R] = f(V_{BN}, V_{BP}, V_{DD}, V_1, V_2, V_3, I) \tag{5.18}$$

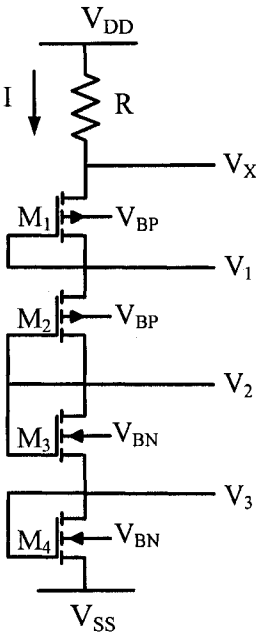


Figure 5.11 Voltage divider designed for 0.18μm technology.

Neural models f_{ann} and $f_{ann,\,5}$ (see (5.1) and (5.2)) are used for various values of I within the given range. Values of W/L for various values of L are calculated for M_1, M_2, M_3 and M_4 and these values along with the value of R calculated are provided as outputs.

$$R = \frac{V_{DD} - V_X}{I} \tag{5.19}$$

A voltage divider is designed for V_{BN} = -1.5 V, V_{BP} = 1.5 V, V_{DD} = 1.2 V, V_1 = 0.6 V, V_2 = 0 V, V_3 = -0.6 V, 100 μA $\leq I \leq$ 200 μA. One of the combinations obtained from the tool and the corresponding simulation verifications are tabulated (see Table 5.12).

Table 5.12 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR VOLTAGE DIVIDER

Device	$V_{SB/BS}$ (Volts)	$V_{SG/GS}$ (Volts)	$V_{SD/DS}$ (Volts)	L (μm)	W/L (μm)	I (Desired) (Amp)	I (Modeled) (Amp)	I (Simulated) (Amp)
$M_{1(PMOS)}$	-0.4	0.5	0.5	0.18	2	100E-6 to 200E-6	180.14E-6 R=550Ω	179.78E-6
$M_{2(PMOS)}$	-0.9	0.6	0.6	0.2	3			
$M_{3(NMOS)}$	-0.9	0.6	0.6	0.4	167			
$M_{4(NMOS)}$	-0.3	0.6	0.6	0.3	5			

5.4.5 Common Drain Amplifier

A typical problem statement for the design of a common drain amplifier is stated as

$$[L, W/L] = f(V_{DD}, V_{BS_{M1}}, V_{BS_{M3}}, V_{GS_{M3}}, A_V, I_{Bias}) \tag{5.20}$$

In this design example, a common drain amplifier is designed for the following user specifications $V_{DD} = -V_{SS} = 1.2$ V, $V_{GS_{M3}} = V_{DS_{M3}} = V_{GS_{M2}} = 0.6$ V, $V_{B_{M1}} = -1.2$ V, $A_v \approx 1$, $V_{BS_{M2}} = V_{BS_{M3}} = 0$ V, and $I_{Bias} \leq 5$ μA. One of the result combinations obtained from the tool and the corresponding simulation verification results from *HSPICE* are tabulated (see Table 5.13). Threshold voltage (V_{th}) for submicron transistors appears to depend not only on V_{BS}, but also on other dc voltages when connected in a totem pole configuration. This may force some transistors of the circuit to operate in regions other than saturation.

Table 5.13 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR COMMON DRAIN AMPLIFIER WITH ACTIVE LOAD

Device	$V_{SB/BS}$ (Volts)	$V_{SG/GS}$ (Volts)	$V_{SD/DS}$ (Volts)	L (μm)	W (μm)	I_{Bias} (Simulated) (Amp)	A_v (Modeled) (V/V)	A_v (Simulated) (V/V)
M_1	-1.2	0.9	1.2	1.2	22.5	1.13E-6	0.91	0.83
M_2	0	0.6	1.2	1.0	22.0	1.13E-6 (Modeled)		
M_3	0	0.6	0.6	1.0	26.0			

5.4.6 Common Source Amplifier

A typical problem statement for the design of a common source amplifier is stated as

$$[L, W/L] = f(V_{DD}, V_{BS_{M1}}, V_{SB_{M2}}, V_{SB_{M3}}, V_{SD_{M3}}, A_V, I_{Bias}) \quad (5.21)$$

In this example, a common source amplifier is designed for the user specifications of $V_{DD} = -V_{SS} = 1.2$ V, $V_{SB_{M2}} = V_{SB_{M3}} = 0$ V, $V_{SG_{M3}} = V_{SD_{M3}} = V_{SG_{M2}} = 0.9$ V, $V_{BS_{M2}} = 0$ V, $|A_V| \geq 30$, and $I_{Bias} = 1.5$ μA. One of the result combinations obtained from the tool and the corresponding simulation verification results obtained from *HSPICE* are tabulated (see Table 5.14).

Table 5.14 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR COMMON SOURCE AMPLIFIER WITH ACTIVE LOAD

Device	$V_{SB/BS}$ **(Volts)**	$V_{SG/GS}$ **(Volts)**	$V_{SD/DS}$ **(Volts)**	L **(μm)**	W **(μm)**	I_{Bias} **Simulated) (Amp)**	A_v **(Modeled) (V/V)**	A_v **(Simulated) (V/V)**
$M_{1(NMOS)}$	0	0.7	1.2	0.8	0.8	1.47E-6		
$M_{2(PMOS)}$	0	0.9	1.2	1.3	113.1	1.47E-6 (Modeled)	−39.3	−35.8
$M_{3(PMOS)}$	0	0.9	0.9	1.3	115.7			

5.4.7 Common Gate Amplifier

A typical problem statement for the design of a common gate amplifier is stated as

$$[L, W/L] = f(V_{DD}, V_{BS_{M1}}, V_{SB_{M2}}, V_{SB_{M3}}, V_{SD_{M3}}, A_V, I_{Bias}) \quad (5.22)$$

In this example, a common gate amplifier is designed for the user specifications of $V_{DD} = -V_{SS} = 1.2$ V, $V_{BS_{M1}} = 0$ V, $V_{SB_{M2}} = V_{SB_{M3}} = 0$ V, $V_{SG_{M3}} = V_{SD_{M3}} = V_{SG_{M2}} = 1.0$ V, $A_v \geq 50$, and $I_{Bias} \leq 15$ μA. One of the result combinations obtained from the tool and the simulation verification results obtained from *HSPICE* are tabulated (see Table 5.15).

Table 5.15 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR COMMON GATE AMPLIFIER WITH ACTIVE LOAD

Device	$V_{SB/BS}$ (Volts)	$V_{SG/GS}$ (Volts)	$V_{SD/DS}$ (Volts)	L (μm)	W (μm)	I_{Bias} (Simulated) (Amp)	A_v (Modeled) (V/V)	A_v (Simulated) (V/V)
$M_{1(NMOS)}$	0	0.7	1.2	1.5	36	10.8E-6	57.99	54.28
$M_{2(PMOS)}$	0	1.0	1.2	0.5	3.5	10.8E-6 (Modeled)		
$M_{3(PMOS)}$	0	1.0	1.0	0.5	4.0			

5.4.8 Push-Pull Amplifier

A typical problem statement for the design of a push-pull amplifier can be stated as

$$[L, W/L] = f(V_{DD}, V_{BS}, V_{SB}, A_v, I) \tag{5.23}$$

A push-pull amplifier is designed in this example for the given user specifications of $V_{DD} = -V_{SS} = 1.2$ V, $|A_v| \geq 400$ V/V, and $I \leq 25$ μA. One of the combinations obtained from the tool and the corresponding *HSPICE* verifications are tabulated (see Table 5.16).

Table 5.16 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR PUSH-PULL AMPLIFIER

Device	$V_{SB/BS}$ (Volts)	$V_{GS/SG}$ (Volts)	$V_{DS/SD}$ (Volts)	L (μm)	W (μm)	$I_{DS/SD}$ (Amp)	Gain (Modeled) (V/V)	Gain (Simulated) (V/V)
$M_{1(NMOS)}$	-0.3	1.2	1.2	1.3	1.3	22.2E-6	– 418.82	– 420.22
$M_{2(PMOS)}$	-0.3	1.2	1.2	1.8	34.2			

5.4.9 Validations for 0.18μm Technology

The current sources/sinks have been tested successfully for numerous values between 2 nA and 28.3 mA for NMOS, and between 82.1 pA and 91.3 mA for PMOS. The current mirrors have shown good agreement for the currents ranging from 5 μA to 150 μA. The amplifier circuits have also been successfully tested for numerous gain values ranging from, 0.9 for the common drain amplifier to 420 for the push-pull amplifier.

5.4.10 Differential Amplifier with Active Current Mirror Load

Differential amplifier is one of the most important design blocks [42] in the design of voltage comparators, operational amplifiers, A/D and D/A converters, *etc.* A popular differential amplifier with active current mirror load is depicted in Fig. 5.12. Small signal equivalent model for the differential amplifier is depicted in Fig. 5.13.

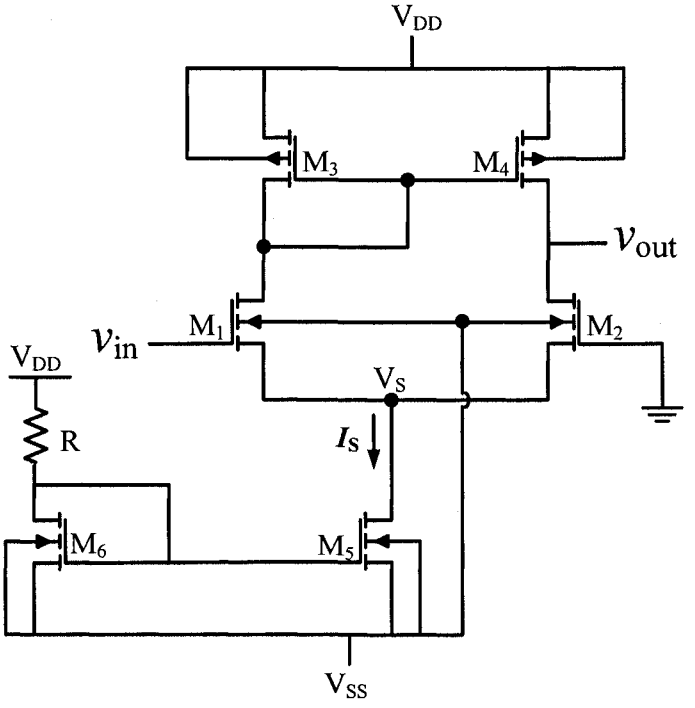


Figure 5.12 A differential amplifier with active current mirror load.

Gain A_v of the differential amplifier shown in Fig. 5.13 is computed using

$$A_v = \frac{v_{out}}{v_{in}} = \frac{g_{m1}}{g_{ds2} + g_{ds4}}. \tag{5.24}$$

A typical problem statement for the design of a differential amplifier can be stated as

$$[L, W/L] = f(V_{DD}, V_S, A_v, I_S) \tag{5.25}$$

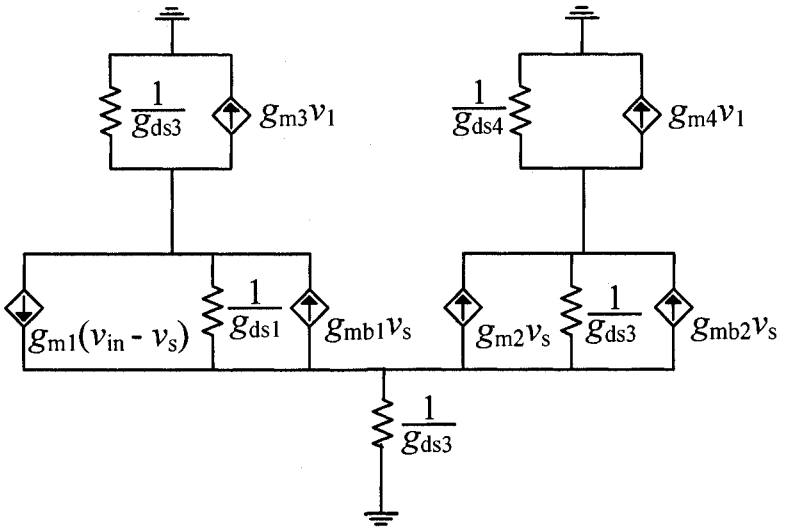


Figure 5.13 An equivalent small signal model for the differential amplifier.

Values of *W/L* for various values of *L* are calculated for M_1, M_2, M_3, M_4, M_5, and M_6 using f_{ann} and $f_{ann,\,5}$ (see (5.1) and (5.2)) are fed to f_{ann2} (see (5.8)) and gain parameters are calculated. Values of *L* and *W/L* satisfying the gain criterion are provided as outputs.

In this example, a differential amplifier with active current mirror load is designed for the following user specifications, V_{DD} = 1.5 V, V_{SS} = -1.5 V, V_S = -0.8 V, $A_V \geq$ 200 V/V, and $I_S \leq$ 15 μA. One of the combinations obtained from the tool and the corresonding *HSPICE* simulation verification results are tabulated (see Table 5.17). The gain curve of the differential amplifier, plotted from *HSPICE* simulation data, is depicted in Fig. 5.14.

Table 5.17 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR A DIFFRENTIAL AMPLIFIER WITH ACTIVE CURRENT MIRROR LOAD

Device	$V_{BS/SB}$ (Volts)	$V_{GS/SG}$ (Volts)	$V_{DS/SD}$ (Volts)	*L* (μm)	*W* (μm)	I_S (Simulated) (Amp)	Gain (Modeled) (V/V)	Gain (Simulated) (V/V)
$M_{1(NMOS)}$ $M_{2(NMOS)}$	-0.7	0.8	1.2	1.7	22.1	10.01E-6 V_S = -0.87V (Simulated)	205.82	201.60
$M_{3(PMOS)}$ $M_{4(PMOS)}$	0	1.1	1.1	0.9	3.6			
$M_{5(NMOS)}$	0	0.8	0.7	1.4	5.6			
$M_{6(NMOS)}$	0	0.8	0.8	1.4	5.6			

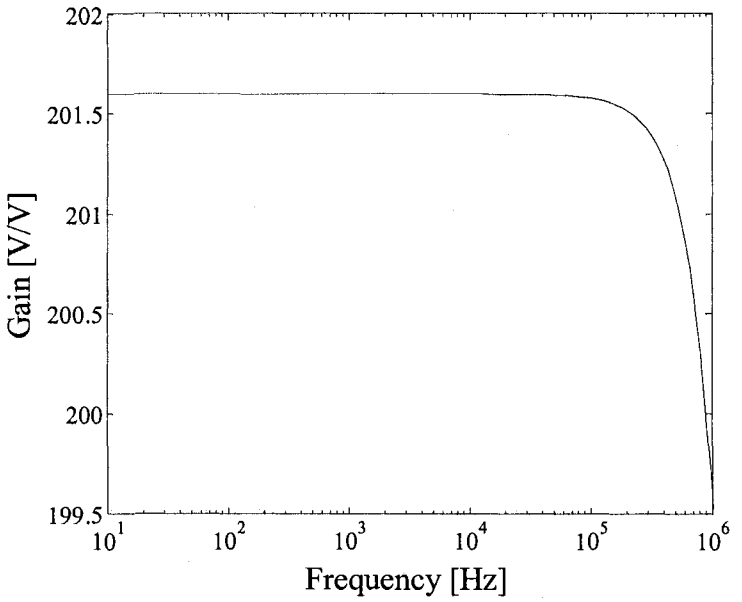


Figure 5.14 Gain curve of the differential amplifier plotted from the *HSPICE* data.

5.4.11 Three Stage Operational Amplifier

Operational amplifier (OPAMP) is the heart of multitude of analog, digital, mixed mode and interface circuits, as such the design of such an integral building block is paramount [23]. A three stage operational amplifier [29] is shown in Fig. 5.15. First stage of the OPAMP is a differential amplifier (see section 5.4.9) that constitutes most of the open loop gain of the OPAMP. Second stage is an intermediate common source amplifier that boosts up the gain and performs DC level shift in order to maintain the DC output voltage of the OPAMP close to zero. Third stage is an output buffer that helps drive resistive load, large capacitive load or combination of both.

Overall open-loop gain (A) of the OPAMP is then defined as the product of gain of all the three stages as

$$A = \frac{v_{\text{out}}}{v_{\text{in}}} = A_1 \times A_2 \times A_3, \tag{5.26}$$

where A_1, A_2, and A_3 are open-loop gains of the differential amplifier, common source amplifier and output buffer respectively and are defined as

$$\text{Differential amplifier gain } A_1 = \frac{v_B}{v_{in}} = \frac{g_{m1}}{g_{ds2} + g_{ds4}}, \tag{5.27}$$

and

$$\text{Common source amplifier gain } A_2 = -\frac{g_{m7}}{g_{ds7} + g_{ds8}}, \tag{5.28}$$

and

$$\text{Output buffer gain } A_3 = \frac{g_{m1}}{g_{m1} + g_{mb1} + g_{ds1} + g_{ds2}}. \tag{5.29}$$

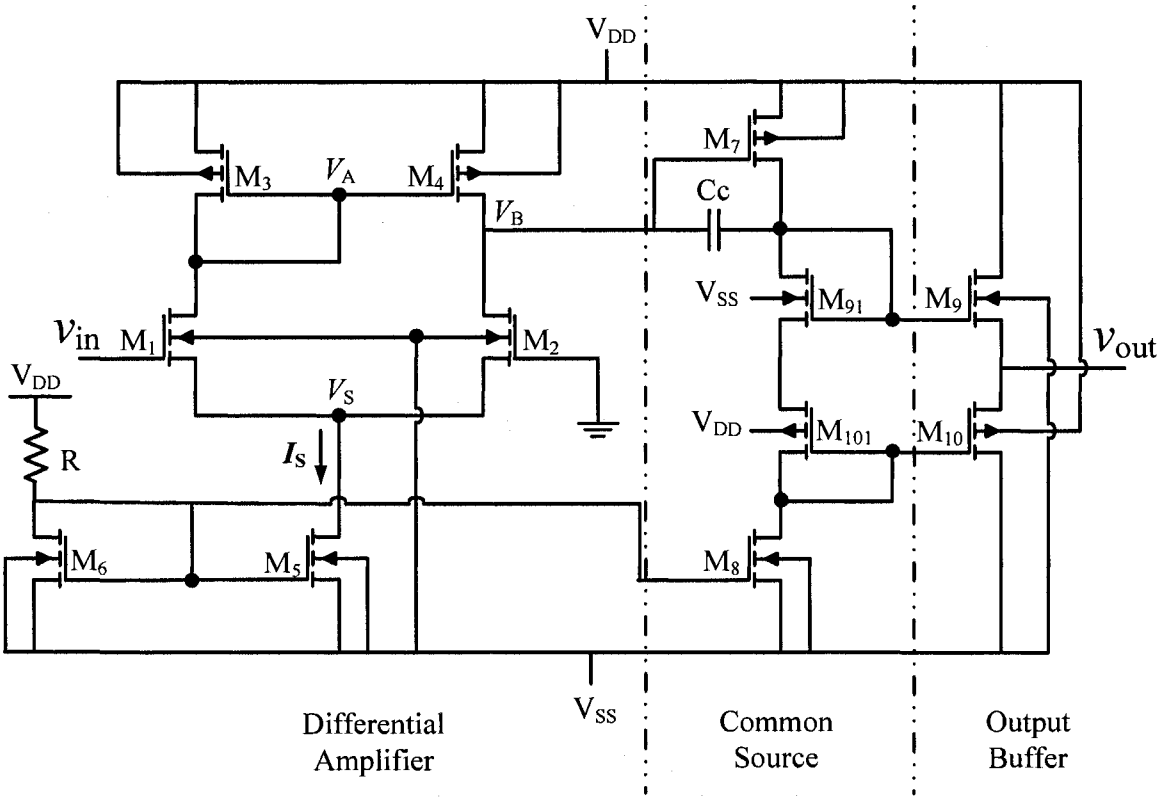


Figure 5.15 A three stage operational amplifier.

Design algorithm for a three stage OPAMP, comprising of three stages, namely differential amplifier, common source amplifier, and output buffer, is discussed in brief.

Phase-1: Differential Amplifier

Step 1: Select differential amplifier bias current I_S based on the user specified power dissipation (P).

Step 2: Select V_{GS} of M_1 and M_2 that satisfy the selected I_S and the source pair dc voltage V_S, specified by the user.

Step 3: Distribute the supply voltage along the differential amplifier, and identify nodal voltages V_S, V_A, and V_B.

Step 4: Design a suitable R, and compute the dimensions of M_5 and M_6 that adhere to prior selected voltages and currents.

Step 5: Finally, design the dimensions of M_1, M_2, M_3, and M_4 that meet the voltage and current values obtained during intermediate processing.

Phase-2: Common Source Amplifier

Step 6: Consider, the biasing current to be the current flowing through M_4, hence copy the dimensions of M_4 to M_7 and design M_8 with current same as that of M_7.

Step 7: Design dimensions of M_{91} and M_{101} satisfying just the current and voltage scheme as they do not constitute the gain (A_2).

Phase-3: Output Buffer

Step 8: Choose suitable dimensions for M_{10} that constitutes to a smaller current as it adds up to the power dissipation.

Step 9: Finally, choose dimensions of M_9 in such a way that DC voltage at v_{out} is maintained close to zero.

A typical problem statement for the design of an operational amplifier is stated as

$$[L, W/L] = f(V_{DD}, A, I_S, V_S, P, V_{out}). \tag{5.30}$$

Operational amplifier is designed for the following user specifications, V_{DD} = 1.5 V, V_{SS} = -1.5 V, $A \geq 75$ dB, $I_S \leq 15$ µA, V_S = -0.8 V, $P \leq 100$ µW and $V_{out} \approx 0$. One of the result combinations obtained from the tool and the corresponding *HSPICE* simulation verification results are tabulated (see Table 5.18). The overall gain curve of the OPAMP plotted from *HSPICE* simulation data is depicted in Fig. 5.16.

Table 5.18 DESIGN PARAMETERS OBTAINED FROM THE TOOL AND THEIR VERIFICATION FROM *HSPICE* FOR A THREE STAGE OPERATIONAL AMPLIFIER

Device	$V_{BS/SB}$ (Volts)	$V_{GS/SG}$ (Volts)	$V_{DS/SD}$ (Volts)	L (µm)	W (µm)	I_{DS} (Modeled) (Amp)	Gain (Modeled) (V/V)	Gain (Simulated) (V/V)
Differential Amplifier Stage								
$M_{1(NMOS)}$ $M_{2(NMOS)}$	-0.7	0.7	1.2	1.7	22.1	5.04E-6	205.82	201.59
$M_{3(PMOS)}$ $M_{4(PMOS)}$	0	1.1	1.1	0.9	3.6	5.04E-6		
$M_{5(NMOS)}$	0	0.8	0.7	1.4	5.6	10.1E-6		
$M_{6(NMOS)}$	0	0.8	0.8	1.4	5.6	10.1E-6		
Common Source Stage								
$M_{7(PMOS)}$	0	1.1	1.1	0.9	3.6	5.11E	73.40	71.19
$M_{8(NMOS)}$	0	0.8	0.6	1.5	3	5.11E		
$M_{91(NMOS)}$	-1.2	0.7	0.7	0.5	29	5.14E		
$M_{101(PMOS)}$	-1.2	0.6	0.6	0.3	17.4	5.12E		
Output Buffer								
$M_{9(NMOS)}$	-1.5	0.4	1.5	0.4	1.2	5.4E-6	0.74	0.76
$M_{10(PMOS)}$	-1.5	0.9	1.5	0.4	4.0	5.45E-6		

The designed value of the resistor $R = (V_{DD} - V_{DS_{M5}})/I_{D_{M5}}$ is 220 KΩ and the dc output voltage V_{out} obtained from the *HSPICE* simulation is 59.6 mV. The power dissipated P is calculated to be 86.049 µW (using eq. 5.31) [39], which successfully satisfies the design criteria for power dissipation.

$$P = (V_{DD} - V_{SS}) \times (I_{M_6} + I_{M_5} + I_{M_8} + I_{M_{10}}). \quad (5.31)$$

The Overall gain of the designed operational amplifier is 80.438dB and fulfills the design requirement for the gain (*i.e.* $\geq$ 75 dB).

In this work, OPAMP is designed for specified gain, supply voltages, dc output, and power dissipation only. As such, frequency compensation and other specifications are not considered. However, robustness of the design and other specifications can be added as the extension of this work.

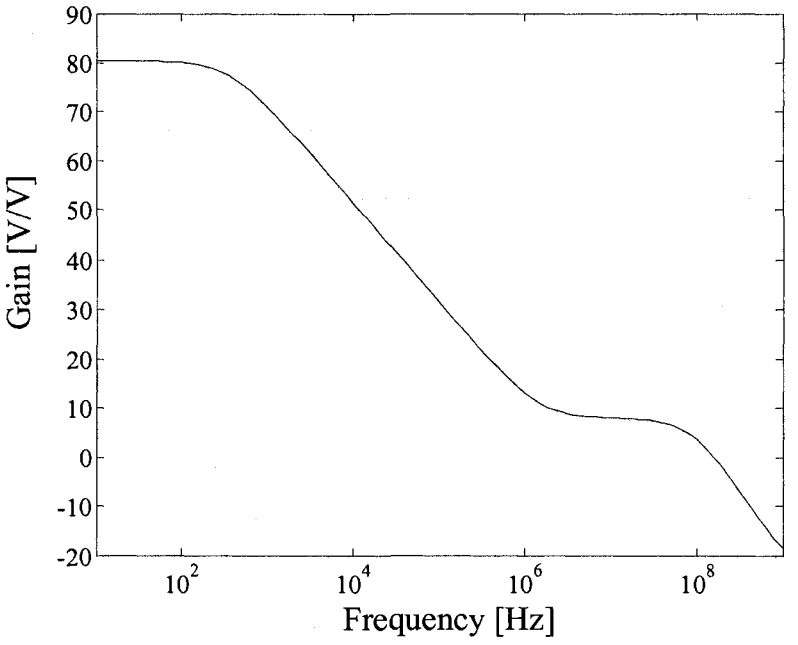


Figure 5.16 Overall gain curve of the operational amplifier plotted from *HSPICE* data.

5.5 SUMMARY

Proposed device modeling algorithms (chapters 3 and 4) are employed for developing accurate and compact MOSFET models from physics based device/circuit simulator data. Various development phases of the design tool, starting with the training data generation, efficient processing of the training data, and modeling of MOSFETs have been briefly

discussed. Applications of the tool for deriving corresponding geometrical dimensions of several MOSFET IC building blocks, given some specifications, in CMOS technologies (0.5μm and 0.18μm) have been presented. The results obtained are verified with *HSPICE* simulations and verifications are results are also tabulated with the results obtained from the design tool.

CHAPTER 6

CONCLUSIONS

6.1 CONTRIBUTIONS

Standard analog design approaches are highly intuitive, time consuming, and demand immense expertise in a multitude of disciplines. First hand design of an analog system is becoming increasingly difficult/impossible because of the myriad of equations involved or innumerable trial and error efforts employing the existing simulation tools. Device modeling is the bottleneck in analog design. As such, accurate device models are critical to development of an efficient analog design tool. In this thesis, two new modeling

algorithms and a new a modeling approach have been introduced to develop simple yet accurate models that lead to an efficient design tool. This thesis is attempted to aid the analog designers by automating the design process with the use of computers, minimizing the human intervention. The tool design process is simple and it can be repeated for any advanced technology by a novice user, without any special expertise.

In this work, several MOSFET modeling algorithms for enhancing the modeling abilities have been presented. Binning algorithm for enhanced modeling with the neural network as a case study has been proposed [43]. This algorithm works on the principle of "Divide and Conquer". A binning parameter is first identified, and the input parameter space is then divided into uniform grids along the selected binning parameter. Several sub-models, one representing each grid, are developed, and then interfaced to generate an overall model. This algorithm helps model the design problem, spanning over wider input parameter space, with simple structured models. This binning concept has been further enhanced to incorporate multi-dimensional modeling problems which helps model the gain of a MOSFET over a wide spectrum of input parameters [44].

Another Neural modeling approach, employing a correction model to improve the accuracy, has been proposed for the first time [45]. In this approach, two neural models namely the desired model (modeled from problem definition), and the correction model (tailored to steer the desired model) are developed as an initial step. Structure of the correction model is not known *apriori*, and is determined in a systematic manner. The prime focus of this approach is to retain the simplicity of the model structure and model training. Both the desired and the correction model are therefore developed employing simple 3-layer MLP networks. This approach helps model the highly nonlinear input

output behaviors accurately, with simple model structure and training method. Modeling process is thus made simple for the novice users. This approach has a potential to be a first-rate optimization technique and has been used to find the geometry of a MOSFET.

Above mentioned algorithms have been put together to develop an analog design tool that offers satisfactorily accurate first-hand design of the commonly used building blocks. Neural networks have been employed for the first time to develop analog building block modules with considerable accuracy. Neural networks exploit the derivative information to model the system under consideration and hence are highly suitable for modeling the continuous spectrum analog circuits. Neural models also require considerably less storage space as compared to the database employed models and makes it easy to maintain and extend the building block module library. Applications of the developed design tool, for specified technologies (*i.e.* 0.5 μm and 0.18 μm), have been presented for several widely used building blocks (*i.e.* from current source/sink to OPAMP level blocks).

6.2 FUTURE WORK

Proposed modeling algorithms and a new neural modeling approach are concepts that are general and can be employed for any other modeling/optimization problem with ease. The systematic approach to first-hand analog circuit design, presented in this work, employs concepts that allow extension of the library, to include modules for novel analog circuits, very easily. The approach can also be easily repeated for the newer technologies without the requisite of detailed knowledge of the discipline. The concept of employing neural networks, for the design/optimization problem, minimizes circuit design as well as optimization times.

In the current version, neural networks are implemented for the modeling purposes that provide substantial improvements in terms of design/optimization time and ease of use. Future work could include the investigation of alternative modeling methods namely, regression modeling, fuzzy logic, and curve fitting, *etc.* Comparison of all these methods will provide substantial information on the best suitable modeling method for the analog integrated circuit design purpose.

In the current version of design tools, modules for commonly used analog building blocks are made available. This allows users to select a building block from the available modules. Future work could include generation of an extensive library of all the building blocks with distinct architectures.

The developed design tools implement Graphical User Interface (GUI) that allow user visualize the circuit being designed/optimized. Output/Design data obtained from the tool is displayed in a tabular format. Future work could include generation of 2-D and 3-D curves of the obtained design from the tool.

As the technology is taking leaps and devices are scaling down to employ lower power designs, Carbon Nano Tubes (CNTs) and silicon nano wires show great prospect in the analog domain. Future work could include incorporation of the CNT and silicon nano wire based building blocks in the design tool, making the design tool completely extensive. This will help to explore novel circuit realization techniques as the design time will be scaled down considerably.

References

[1] B.A.A. Antao, "Trends in CAD of Analog ICs," *IEEE Circuits and Devices Magazine,* vol. 12, pp. 31-41, 1996.

[2] C. Toumazou and C.A. Makris, "Automated circuit generation: New concepts and methods," *IEEE Trans. Computer-Aided Design*, vol. 14, 1995.

[3] D.D. Gajski, N.D. Dutt, and B.M. Pangrale, "Silicon compilation (tutorial)," *Proc. IEEE CICC*, Rochester, NY, May 1986, pp. 102-110.

[4] R.E. Massara, Y. Nadiadi, and G.L. Winder, "Silicon compilation in analog and digital custom VLSI design," *Progress in Computer Aided VLSI Design*, New York, NY: Ablex, 1989, pp. 91-135.

[5] J.L. Castro, C.J. Mantas, and J.M. Benitez, "Interpretation of artificial neural networks by means of fuzzy rules," *IEEE Trans. Neural Networks*, vol. 13, pp. 101-116, 2002.

[6] Y.P. Tsividis, "Analog MOS integrated circuits-certain new ideas, trends, and obstacles," *IEEE J. Solid-State Circuits*, vol. 22, no. 3, pp. 317-321, 1987.

[7] W. Nye, D.C. Riley, A. Sangiovanni-Vincentelli, and A.L. Tits, "DELIGHT.SPICE: an optimization-based system for the design of integrated circuits," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 501-519, 1988.

[8] J.M. Shyu, A. Sangiovanni-Vincentelli, "ECSTASY: a new environment for IC design optimization," *IEEE Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 1988, pp. 484-487.

[9] J.C. Lai, J.S. Kueng, H.C. Chen, and F.J. Fernandez, "ADOPT-A CAD tool for analog circuit design," *IEEE Circuits and Devices Magazine,* vol. 4, pp. 29-30, 1988.

[10] P. E. Allen, "A tutorial--computer aided design of analog integrated circuits," *IEEE Proc. Custom Integrated Circuits Conf.*, Rochester, NY, May 1986, pp. 608-616.

[11] D.C. Stone, J.E. Schroeder, R.H. Kaplan, and A.R. Smith, "Analog CMOS building blocks for custom and semicustom applications," *IEEE Trans. Electron Devices*, vol. 31, pp. 189-195, 1984.

[12] T. Pletersek, J. Trontelj, L. Trontelj, I. Jones, and G. Shenton, "High-performance designs with CMOS analog standard cells," *IEEE J. Solid-State Circuits*, vol. 21, pp. 215-222, 1986.

[13] M.J.S. Smith, C. Portmann, C. Anagnostopoulos, P.S. Tschang, R. Rao, P. Valdenaire, and H. Ching, "Cell libraries and assembly tools for analog/digital CMOS and BiCMOS application-specific integrated circuit design," *IEEE J. Solid-State Circuits*, vol. 24, pp. 1419-1432, 1989.

[14] P.E. Allen and E. R. Macaluso, "AIDE2: an automated analog IC design," *IEEE Proc. Custom Integrated Circuits Conf.*, Portland, OR, May 1985, pp. 498-501.

[15] J.M. Cohn, D.J. Garrod, R.A. Rutenbar, and R. Carley, *Analog Device-Level Layout Automation*, New York, NY: Springer, 1994.

[16] A.L. Ressler, "A circuit grammar for operational amplifier design," Ph.D. dissertation, MIT, 1984.

[17] R.J.B. Bowman and D.J.L. Lane, "A knowledge-based system for anlog integrated circuit design," *IEEE Proc. Int. Conf. Computer-Aided Design*, Santa Clara, CA, May 1985, pp. 210-212.

[18] F. El-Turky, and E.E. Perry, "BLADES: an artificial intelligence approach to analog circuit design," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 680-692, 1989.

[19] R. Harjani, R.A. Rutenbar, and L.R. Carley, "OASYS: a framework for analog circuit synthesis," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 1247-1266, 1989.

[20] E. Berkcan, and F. Yassa, "Towards mixed analog/digital design automation: a review," *IEEE Int. Symp. Circuits Sys.*, New Orleans, LA, May 1990, pp. 809-815.

[21] M.G.R. Degrauwe, B.L.A.G. Goffart, C. Meixenberger, M.L.A. Pierre, J.B. Litsios, J. Rijmenants, O.J.A.P. Nys, E. Dijkstra, B. Joss, M.K.C.M. Meyvaert, T.R. Schwarz, and M.D. Pardoen, "Towards an analog system design environment," *IEEE J. Solid-State Circuits*, vol. 24, pp. 659-671, 1989.

[22] M.G.R. Degrauwe, O. Nys, E. Dijkstra, J. Rijmenants, S. Bitz, B.L.A.G. Goffart, E.A. Vittoz, S. Cserveny, C. Meixenberger, G. van der Stappen, and H.J. Oguey, "IDAC: an interactive design tool for analog CMOS circuits," *IEEE J. Solid-State Circuits*, vol. 22, pp. 1106-1116, 1987.

[23] H.Y. Koh, C.H. Sequin, and P.R. Gray, "OPASYN: a compiler for CMOS operational amplifiers," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 113-125, 1990.

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

[25] G.G.E. Gielen, H.C.C. Walscharts, and W.M.C. Sansen, "Analog circuit design optimization based on symbolic simulation and simulated annealing," *IEEE J. Solid-State Circuits*, vol. 25, pp. 707-713, 1990.

[26] A.H. Fung, B.W. Lee, and B.J. Sheu, "Self-reconstructing technique for expert system-based analog IC designs," *IEEE Trans. Circuits and Systems*, vol. 36, pp. 318-321, 1989.

[27] B.J. Sheu, J.C. Lee, and A.H. Fung, "Flexible architecture approach to knowledge-based analogue IC design," *Proc. G IEEE Circuits, Devices and Systems*, vol. 137, pp. 266-274, 1990.

[28] D. Johns, and K. Martin, *Analog Integrated Circuit Design*, New York, NY: John Wiley and Sons, 1997.

[29] R. Raut, *Introduction to Analog VLSI - Lecture Notes*, Montreal, QC: Concordia University Publication, 2002.

[30] Q.J. Zhang, K.C. Gupta, and V.K. Devabhaktuni, "Artificial neural networks for RF and microwave design–From theory to practice," *IEEE Trans. Microwave Theory Tech.*, vol. 51, pp. 1339-1350, 2003.

[31] V.K. Devabhaktuni, M.C.E. Yagoub, Y. Fang, J. Xu, and Q.J. Zhang, "Neural networks for microwave modeling: Model development issues and nonlinear modeling techniques," *Int. J. RF Microwave Computer-Aided Eng.*, vol. 11, pp. 4-21, 2001.

[32] X. Ding, V.K. Devabhaktuni, B. Chattaraj, M.C.E. Yagoub, M. Deo, J. Xu, and Q.J. Zhang "Neural-network approaches to electromagnetic-based modeling of passive components and their applications to high-frequency and high-speed nonlinear circuit optimization," *IEEE Trans. Microwave Theory Tech.*, vol. 52, pp. 436-449, 2004.

[33] V.K. Devabhaktuni, B. Chattaraj, M.C.E. Yagoub, and Q.J. Zhang, "Advanced microwave modeling framework exploiting automatic model generation, knowledge neural networks, and space mapping," *IEEE Trans. Microwave Theory Tech.*, vol. 51, pp. 1822-1833, 2003.

[34] P.M. Watson, K.C. Gupta, and R.L. Mahajan, "Development of knowledge based artificial neural network models for microwave components," *Proc. IEEE Int. Microwave Symp.*, Baltimore, MD, June 1998, pp. 9-12.

[35] *NeuroModeler*, Q.J. Zhang, Dept. of Electronics, Carleton University, 1125 Colonel By Drive, Ottawa, K1S 5B6, Ontario, Canada.

[36] E. Abou-Allam and T. Manku, "A small-signal MOSFET model for radio frequency IC applications," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 437-447, 1997.

[37] P.R. Gray, P.J. Hurst, S.H. Lewis, and R.G. Meyer, *New Analysis and Design of Analog Integrated Circuits*, New York, NY: John Wiley and Sons, 2001.

[38] A. Zeki, and H. Kuntman, "Accurate and high output impedance current mirror suitable for CMOS current output stages," *Electronics Letters*, vol. 33, pp. 1042-1043, June 1997.

[39] E. Sackinger and W. Guggenbuhl, "A high swing, high impedance MOS cascode circuit," *IEEE J. Solid-State Circuits*, vol. 25, pp. 289-297, Feb. 1990.

[40] R.J. Baker, H.W. Li, and D.E. Boyce, *CMOS Circuit Design, Layout, and Simulation*, New York, NY: IEEE Press, 1998.

[41] P.E. Allen, and D.R. Holberg, *CMOS Analog Circuit Design*, 198 Madison Avenue, NY: Oxford University Press, 2002.

[42] B. Razavi, *Design of Analog CMOS Integrated Circuits*, New York, NY: McGraw-Hill, 2001.

[43] K.A. Mendhurwar, V.K. Devabhaktuni, and R. Raut, "Binning algorithm for accurate computer aided device modeling," *IEEE Int. Symp. Circuits Systems*, Seattle, WA, May 2008 (Accepted).

[44] K.A. Mendhurwar, V.K. Devabhaktuni, and R. Raut, "A new computer-aided multi-dimensional device modeling algorithm based on binning concepts," *IEEE Int. conference on Microelectronics*, Nis, Serbia, May 2008 (Accepted).

[45] K.A. Mendhurwar, V.K. Devabhaktuni, and R. Raut, "A new neural network modeling approach based on a correction model concept," *IEEE Microwave and Wireless Components Letters* (Submitted).

APPENDIX-A

Application of Binning Concepts Based Modeling Algorithms to a Passive Component

In this appendix, modeling algorithms based on binning concepts, proposed in chapter 3 of this thesis, are employed successfully to modeling of passive component (*i.e.* spiral inductor). Illustration examples confirm the validity of the proposed modeling algorithms in passive domain.

1. Illustration Examples

1.1. Single Dimensional Example

Here, a spiral inductor offering the benefit of compact size is modeled. The modeling problem can be stated as

$$S_{21,\,\mathrm{dB}} = f(W,\, S,\, d_{\mathrm{in}},\, freq), \tag{1}$$

where W, S and d_{in} represent line width, spacing between lines, and inner diameter of the inductor (see Fig. A.1(a)). It is to be noted that *freq* denotes frequency and $S_{21,\,\mathrm{dB}}$ is magnitude of S_{21} in dB. Number of turns $N = 5$ is fixed. Training data for modeling the spiral inductor is obtained using Ansoft *HFSS*. A datasheet of S_{21} is generated by sweeping parameters W, S, d_{in} and *freq* along uniform grids. For example, training data corresponding to $W = 12\mu\mathrm{m}$ and $S = 2\mu\mathrm{m}$ is shown in Fig. A.1(b).

First, a neural model with four inputs (W, S, d_{in}, *freq*) and one output ($S_{21,\,dB}$) is developed using the standard approach. *NeuroModeler* used for training, resulted in a best possible neural model. Model responses corresponding to the training data of Fig. A.1(b) are shown in Fig. A.1(c). In an attempt to closely inspect the neural model, model errors for different sub-ranges along the W axis are computed (see Table A.2). The average error of the neural model is greater than 5.83%, and the worst-case performance of the model is unacceptable.

For the spiral inductor, first-hand analysis equations are not readily available. An alternative way of computing derivative information involves - (i) finding the transfer function of the lumped model of the inductor, (ii) expressing lumped elements in the transfer function in terms of design parameters, and (iii) deriving analytical expressions. Such an approach is tedious and sensitivity analysis is impractical. As such, the proposed binning algorithm based on trial-and-error is applied. Internal diameter d_{in} is ruled out due to its wider range. Other inputs are considered as binning parameters, and the corresponding aggregate errors are evaluated (see Table A.1). Among other parameters, W offers better results, and is selected as the binning parameter. It is removed from $\boldsymbol{x}$, and the subspace is divided into 7 intervals using uniform-grids along W axis. Neural models for all the 7 intervals are developed and are interfaced to create an overall model. As seen in Table A.2, average and worst-case errors using the proposed algorithm are less than 0.9% and 8% respectively. Proposed model responses for W = 12μm and S = 2μm are shown in Fig 4(d).

TABLE A.1

COMPARISION OF MODEL ACCURACIES LEADING TO SELECTION OF THE BINNING PARAMETER

Binning Parameter	Percentage Aggregate Error Measure	
	Average Error	*Worst-Case Error*
W	3.56%	17.99%
S	8.64%	85.98%
Freq	17.28%	155.67%

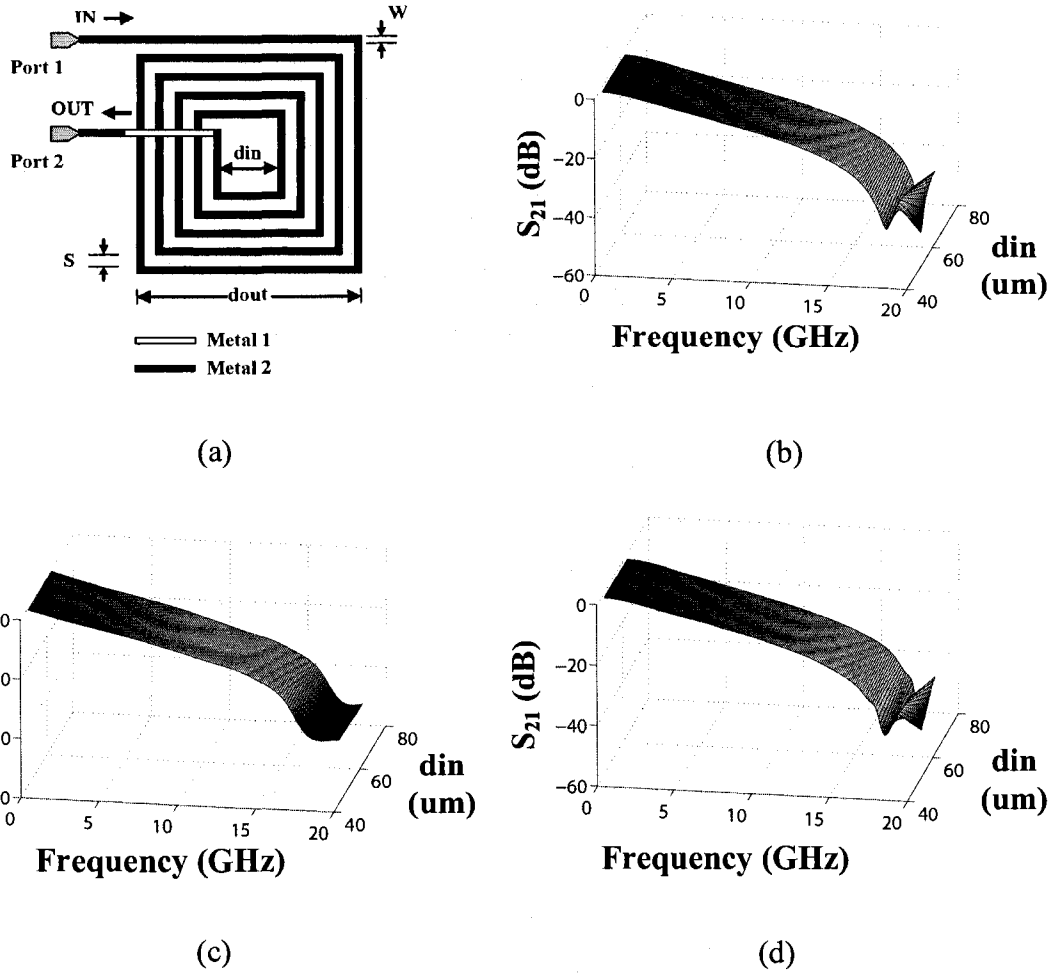


Figure A.1 (a) Geometry of a square spiral inductor, (b) Training data corresponding to $W = 12\mu m$ and $S = 2\mu m$, (c) Neural model responses using the standard approach, and (d) Neural model responses using the proposed single dimensional binnning algorithm.

TABLE A.2

COMPARISION OF MODEL ACCURACIES FOR SINGLE DIMENSIONAL SPIRAL INDUCTOR EXAMPLE

W Range (in um)	Neural Model Using The Standard Approach		Overall Model Using The Proposed Algorithm	
	Average Error	*Worst-Case Error*	*Average Error*	*Worst-Case Error*
1-3	6.31%	244.64%	0.56%	7.99%
3-5	5.83%	122.74%	0.87%	6.53%
5-7	8.05%	169.12%	0.69%	6.51%
7-9	8.50%	180.35%	0.77%	7.54%
9-11	8.25%	287.26%	0.53%	6.93%
11-13	9.20%	287.26%	0.80%	6.11%
13-15	9.14%	267.34%	0.64%	7.25%

1.2 Multi Dimensional Example

Here, *S*-prameters of a spiral inductor offering compact size are modeled. Modeling problem can be stated as

$$[S_{11,\,\mathrm{dB}},\ S_{21,\,\mathrm{dB}}] = \boldsymbol{f}(W,\ S,\ d_{\mathrm{in}},\ freq), \tag{2}$$

where W, S and d_{in} represent line width, spacing between lines, and inner diameter of the spiral inductor (Fig. A.2(a)). It is to be noted that *freq* denotes frequency and $S_{11,\,\mathrm{dB}}$ and $S_{21,\,\mathrm{dB}}$ are magnitudes of S_{11} and S_{21} in dB. Number of turns N = 4.5 is kept constant. Training data for modeling the spiral inductor is obtained using Ansoft *HFSS*. A datasheet of S_{11} and S_{21} is generated by sweeping W, S, d_{in} and *freq* along uniform grids. For example, training data for $S_{21,\,\mathrm{dB}}$ when W = 5μm and S = 2μm is shown in Fig. A.2(b).

First, a neural model with four inputs (W, S, d_{in}, *freq*) and two outputs ($S_{11,\,\mathrm{dB}}$, $S_{21,\,\mathrm{dB}}$) is developed using the standard approach. *NeuroModeler* used for training, resulted in a

best possible neural model. Model responses corresponding to the training data of Fig. A.2(b) are shown in Fig. A.2(c). In an attempt to closely inspect the neural model, model errors for different sub-ranges along the W axis are computed (see Table A.3). Average errors of the neural model are greater than 3.82%, and worst-case performance of the model is unacceptable.

For spiral inductor, sensitivity analysis is impractical [43]. As such, in the proposed multi-dimensional binning algorithm, impact indices are estimated, using the training data, to identify the binning parameter. All the physical input parameters are considered as potential binning parameters, and the corresponding impact indices and standard deviations are evaluated (Table A.3). Among other parameters, W offers better results, and is selected as the binning parameter. It is then removed from $\boldsymbol{x}$, and the subspace is divided into 7 intervals using uniform-grids along W axis. Neural sub-models for all the 7 intervals are developed and are interfaced into an overall model.

As seen in Table A.4, average and worst-case errors of the overall model obtained using the proposed multi-dimensional binning algorithm are less than 1% and 5% respectively. Proposed neural model responses corresponding to W = 5μm and S = 2μm are shown in Fig A.2(d).

TABLE A.3 STATISTICAL DATA FOR SELECTION OF THE BINNING PARAMETER

Binning Parameter	Standard Deviation	Impact Index	
		S_{11}	S_{21}
S	2.78	0.09	0.39
W	4.75	0.36	0.43
d_{in}	31.2	0.29	0.53

TABLE A.4
COMPARISON OF MODEL ACCURACIES FOR MULTI DIMENSIONAL SPIRAL INDUCTOR EXAMPLE

W Range (μm)	Neural Model Using The Standard Approach				Overall Model Using The Proposed Algorithm			
	Average Error		*Worst-Case Error*		*Average Error*		*Worst-Case Error*	
	S_{11}	S_{21}	S_{11}	S_{21}	S_{11}	S_{21}	S_{11}	S_{21}
1-3	68.8%	3.86%	258.8%	515.6%	0.94%	0.59%	4.98%	4.64%
3-5	271.9%	7.86%	187.14%	96.18%	0.81%	0.95%	4.86%	4.55%
5-7	11.26%	6.69%	45.12%	104.48%	0.86%	0.37%	4.55%	4.50%
7-9	7.62%	6.76%	39.40%	85.36%	0.58%	0.50%	2.62%	4.89%
9-11	5.15%	7.83%	17.83%	113.04%	0.61%	0.60%	3.08%	4.58%
11-13	3.82%	6.46%	12.49%	99.35%	0.61%	0.54%	3.08%	4.36%
13-15	3.84%	7.60%	10.94%	150.72%	0.88%	0.70%	4.19%	4.98%

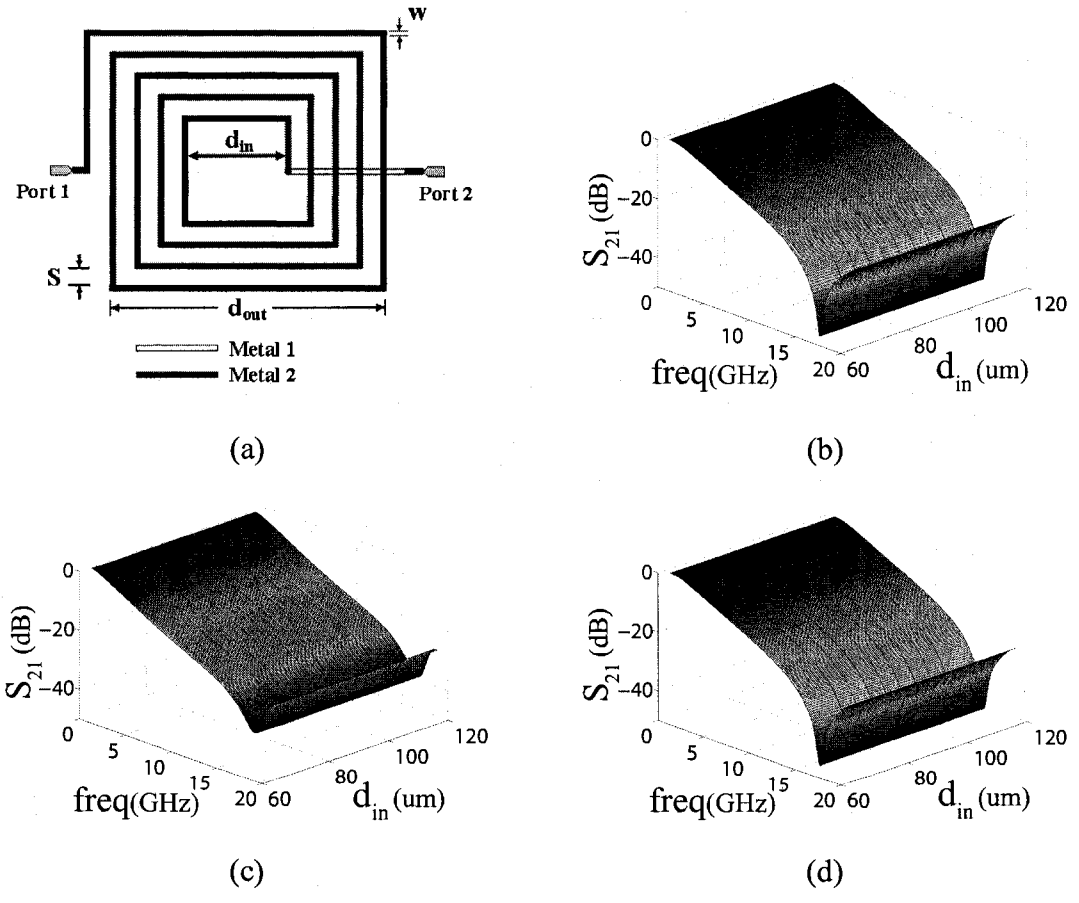


Fig. A.2 (a) Geometry of a square spiral inductor, (b) Training data corresponding to $W = 5\mu m$ and $S = 2\mu m$, (c) Neural model responses using the standard approach, and (d) Neural model responses using the proposed multi-dimensional binning algorithm.

APPENDIX-B

Application of Correction Model Based Neural Modeling Approach to a Passive Component

In this appendix, neural modeling approach, proposed in chapter 4 of this thesis, is extended successfully to modeling of passive component (*i.e.* spiral inductor). Example confirms the usability of the proposed approach in passive domain.

On-chip passive component models (*e.g.* spiral inductor models) are critical to RFIC design and optimization. A compact ANN model of an on-chip spiral inductor is to be developed *i.e.*

$$d_{\text{in}} = f_{\text{ann}}(W, L, freq), \tag{1}$$

where d_{in} and W represent inner diameter and line width (see Fig. B.1(a)), L represents inductance, and *freq* denotes frequency. Keeping the number of turns $N = 4.5$ and spacing between lines S = 5μm fixed, training data is generated using CPU-expensive full-wave 3D simulations of an electromagnetic (EM) solver, namely Ansoft *HFSS*, by varying d_{in}, W, and *freq* along uniform grids. For example, training data corresponding to W = 1μm is shown in Fig. B.1(b).

First, desired model f_{ann} with three inputs and one output (d_{in}) is trained using *NeuroModeler*. f_{ann} responses corresponding to training data of Fig. B.1(b) are shown in Fig. B.1(c). Assuming E_{user} = 5%, this stand-alone model exhibits unacceptable E and E_{worst} (>10% and >221%). Of the three potential correction models,

$$L = f_{ann,2}(W, d_{in}, freq) \tag{2}$$

is selected as the correction model based on (4.6) and Table B.1. Finally, f_{ann} is used together with $f_{ann,2}$ thereby resulting in an acceptable model. Initial value of Δy is set to be 5μm during model utilization, and iteratively updated as in the pseudocode. Improved responses for W = 1μm are shown in Fig B.1(d). As can be seen in Table B.2, E and E_{worst} based on the proposed approach are significantly improved (<1% and <4% respectively).

TABLE B.1
COMPARISON OF MODEL ACCURACIES FOR THE SPIRAL INDUCTOR EXAMPLE

Error	**Stand-Alone Desired Model**	**Potential Correction Models**			**Proposed Approach**
		L	*W*	*freq*	
E	10.4%	0.95%	13.38%	8.56%	0.98%
E_{worst}	221.45%	3.98%	73.96%	56.8%	3.87%

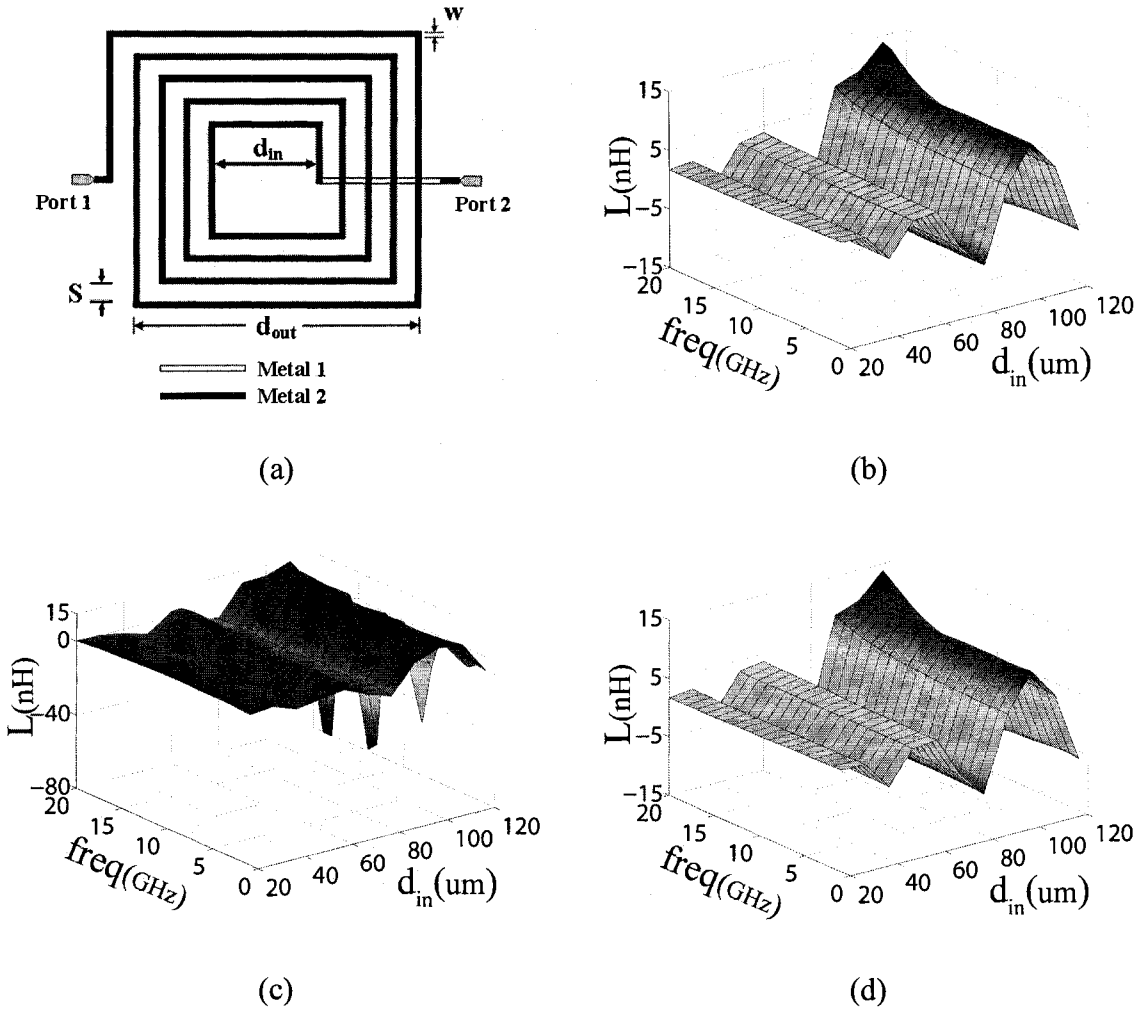


Fig. B.1 (a) Geometry of a square spiral inductor, (b) Training data corresponding to $W =$ 1μm, (c) Responses of the stand-alone neural model based on the standard approach, and (d) Responses of the proposed neural model.

APPENDIX-C

Screenshots of the Single Stage Amplifier design module of the developed tool

1. Insert the DVD of the designed tool.

2. Explore the DVD and double click CAD_Tool.exe

3. CAD tool will start with the screen shown below. One of the design modules is shown.

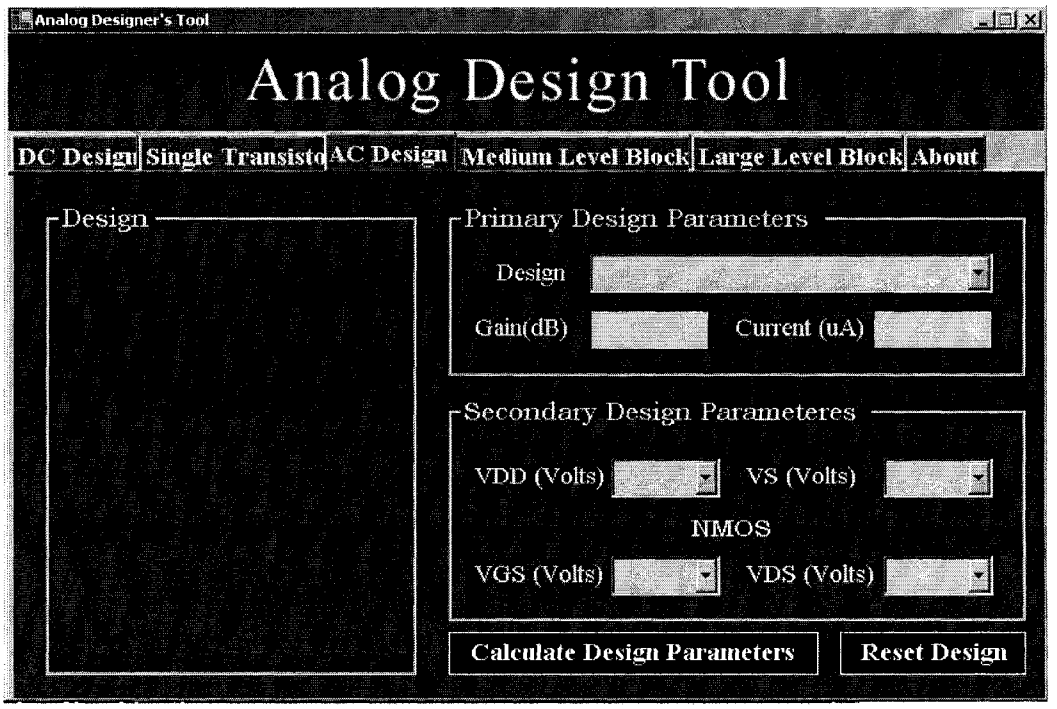


Figure C.1 Single stage amplifier designing module.

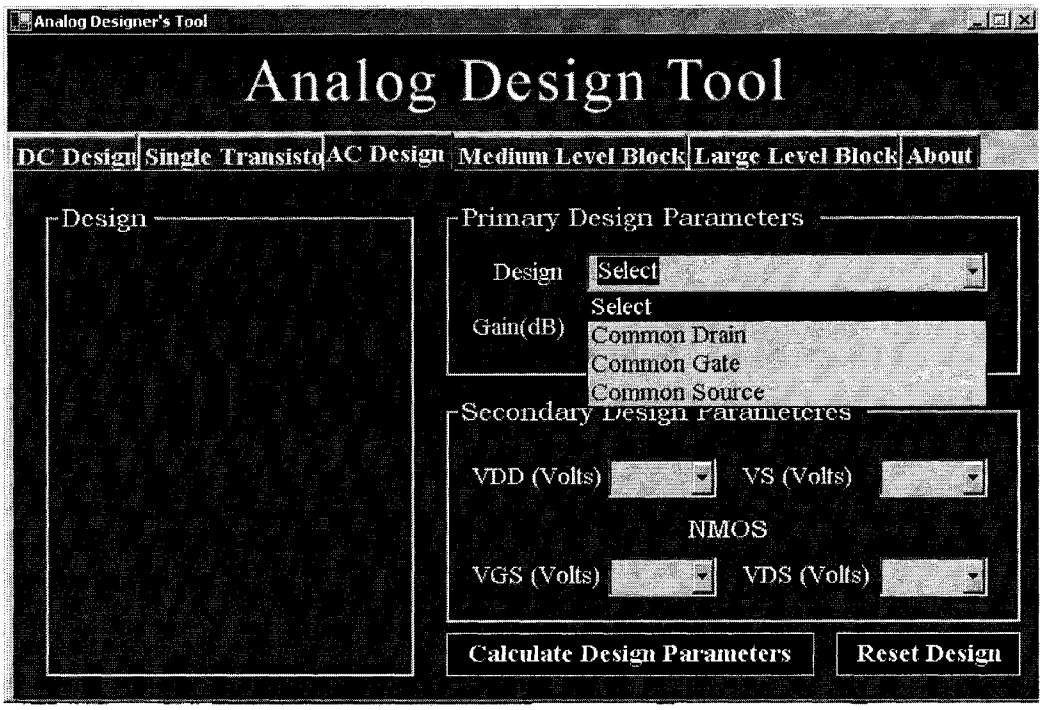


Figure C.2. Design architectures available at the moment.

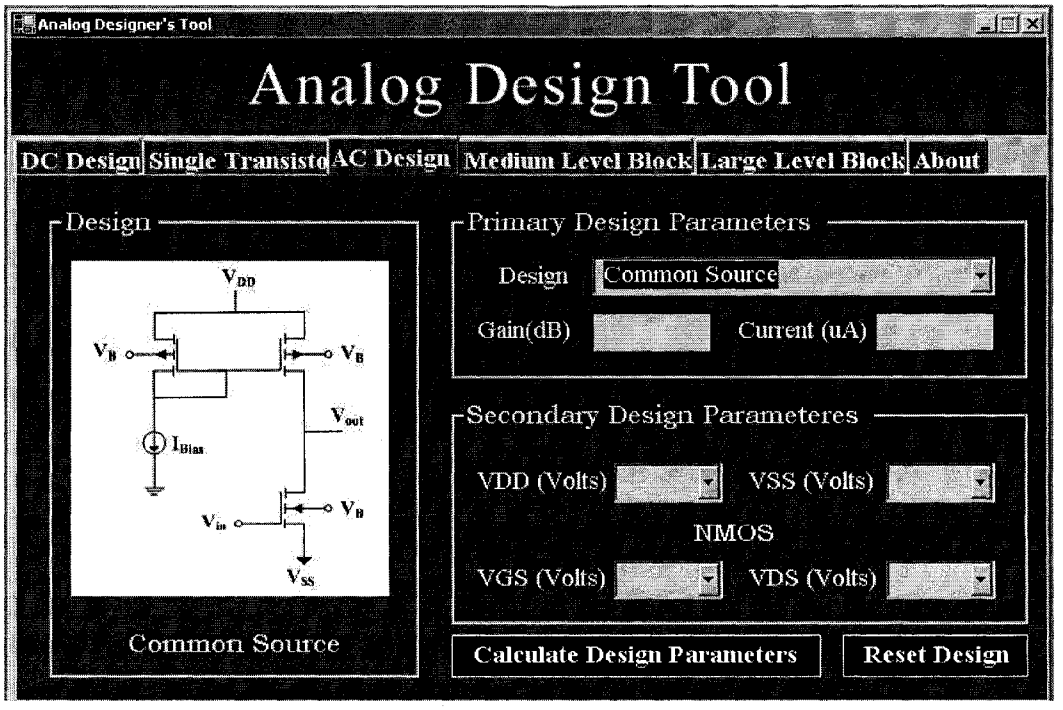


Figure C.3. Selection of the design architecture.

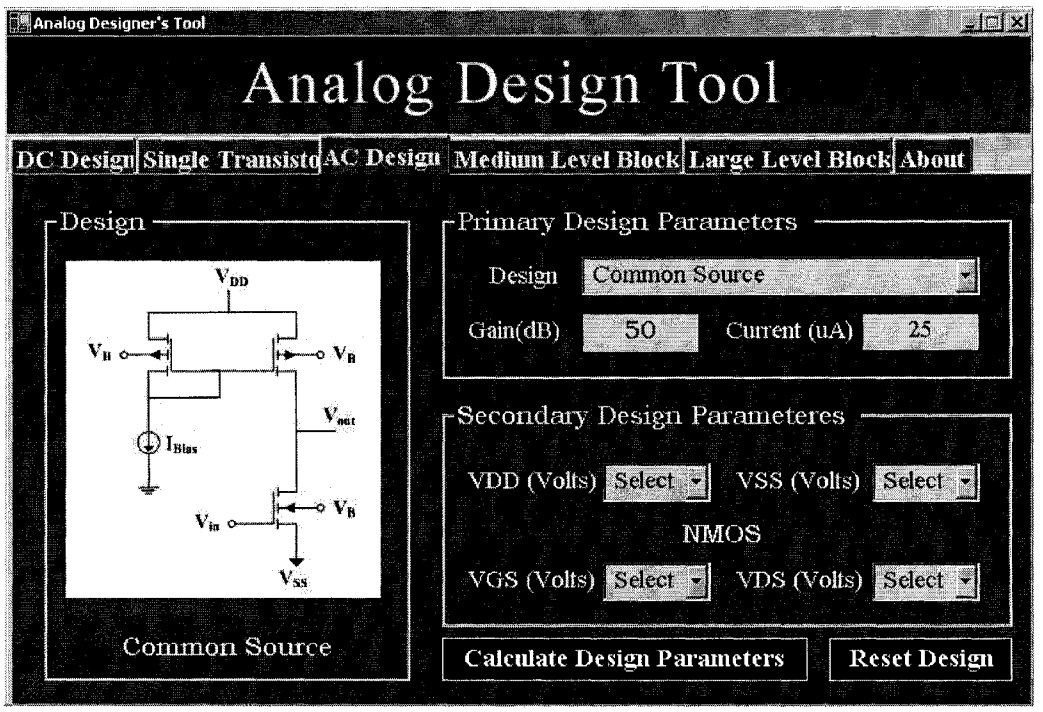


Figure C.4. Selection of the primary design parameters.

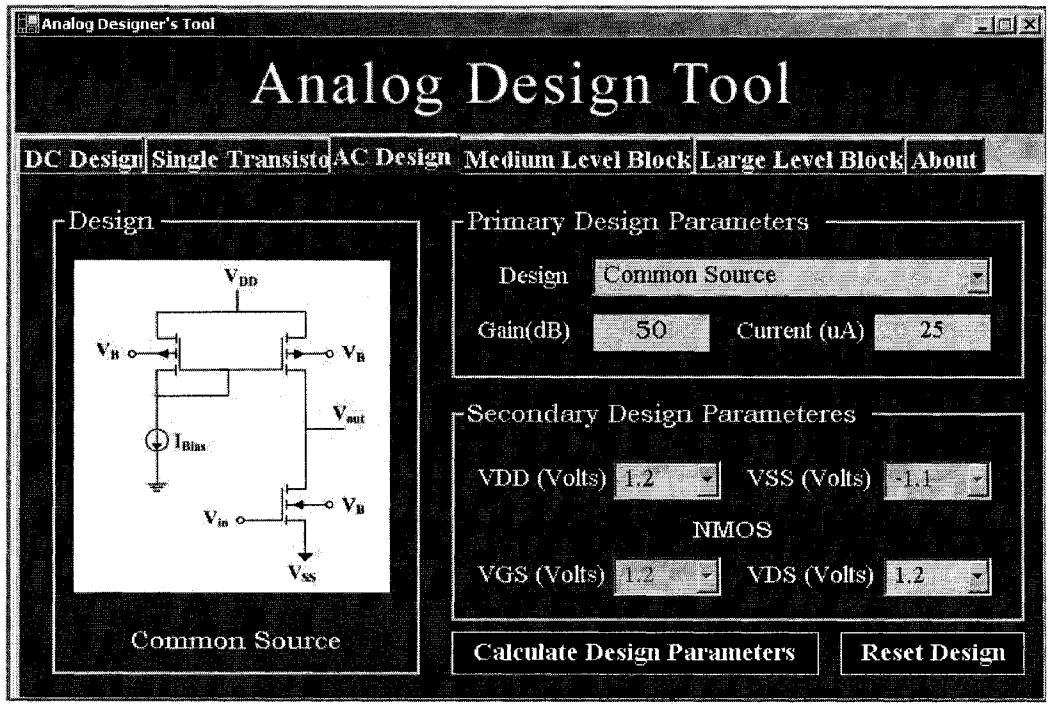
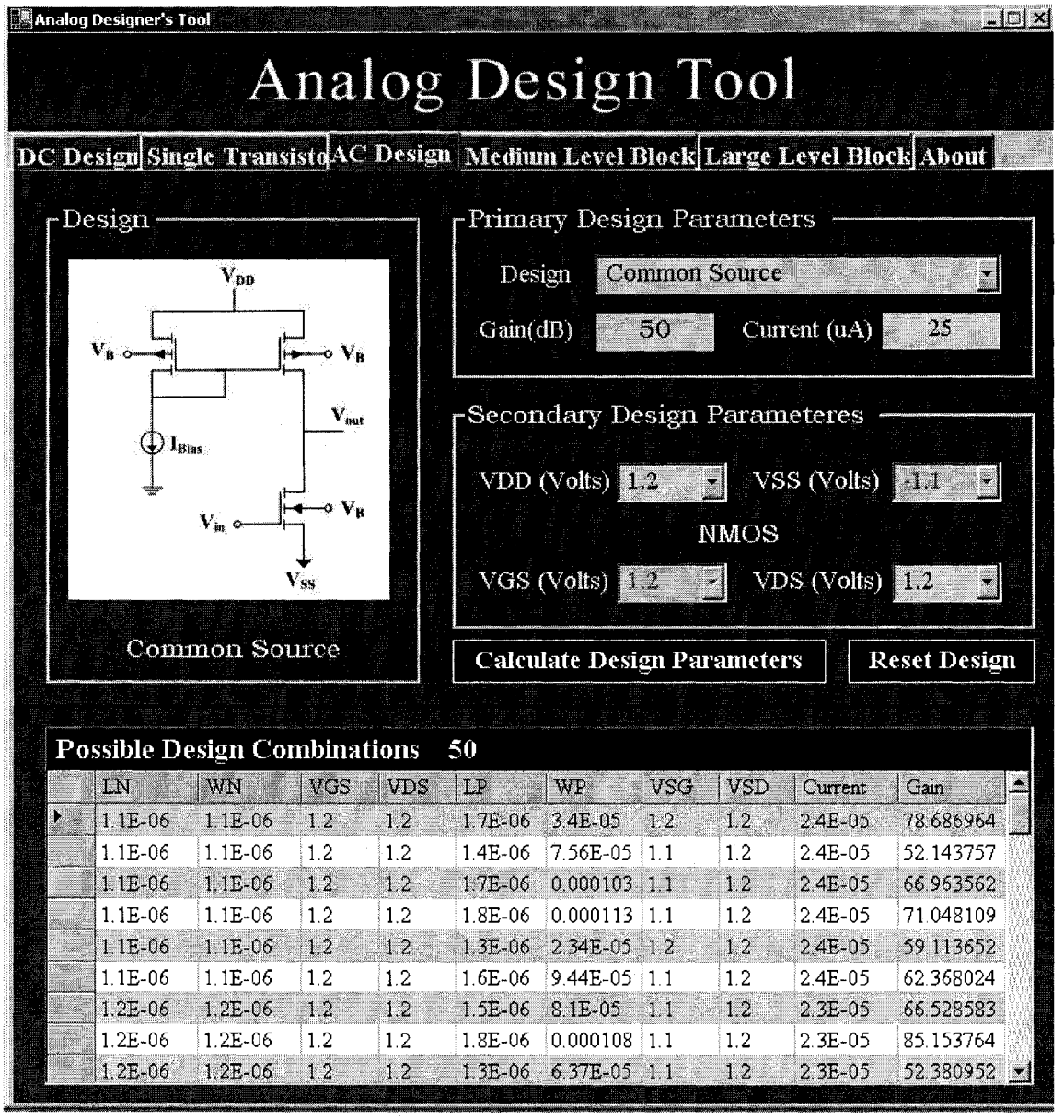


Figure C.5. Selection of the secondary design parameters.



LN	WN	VGS	VDS	LP	WP	VSG	VSD	Current	Gain
1.1E-06	1.1E-06	1.2	1.2	1.7E-06	3.4E-05	1.2	1.2	2.4E-05	78.686964
1.1E-06	1.1E-06	1.2	1.2	1.4E-06	7.56E-05	1.1	1.2	2.4E-05	52.143757
1.1E-06	1.1E-06	1.2	1.2	1.7E-06	0.000103	1.1	1.2	2.4E-05	66.963562
1.1E-06	1.1E-06	1.2	1.2	1.8E-06	0.000113	1.1	1.2	2.4E-05	71.048109
1.1E-06	1.1E-06	1.2	1.2	1.3E-06	2.34E-05	1.2	1.2	2.4E-05	59.113652
1.1E-06	1.1E-06	1.2	1.2	1.6E-06	9.44E-05	1.1	1.2	2.4E-05	62.368024
1.2E-06	1.2E-06	1.2	1.2	1.5E-06	8.1E-05	1.1	1.2	2.3E-05	66.528583
1.2E-06	1.2E-06	1.2	1.2	1.8E-06	0.000108	1.1	1.2	2.3E-05	85.153764
1.2E-06	1.2E-06	1.2	1.2	1.3E-06	6.37E-05	1.1	1.2	2.3E-05	52.380952

Figure C.6. Results displayed from the developed tool.

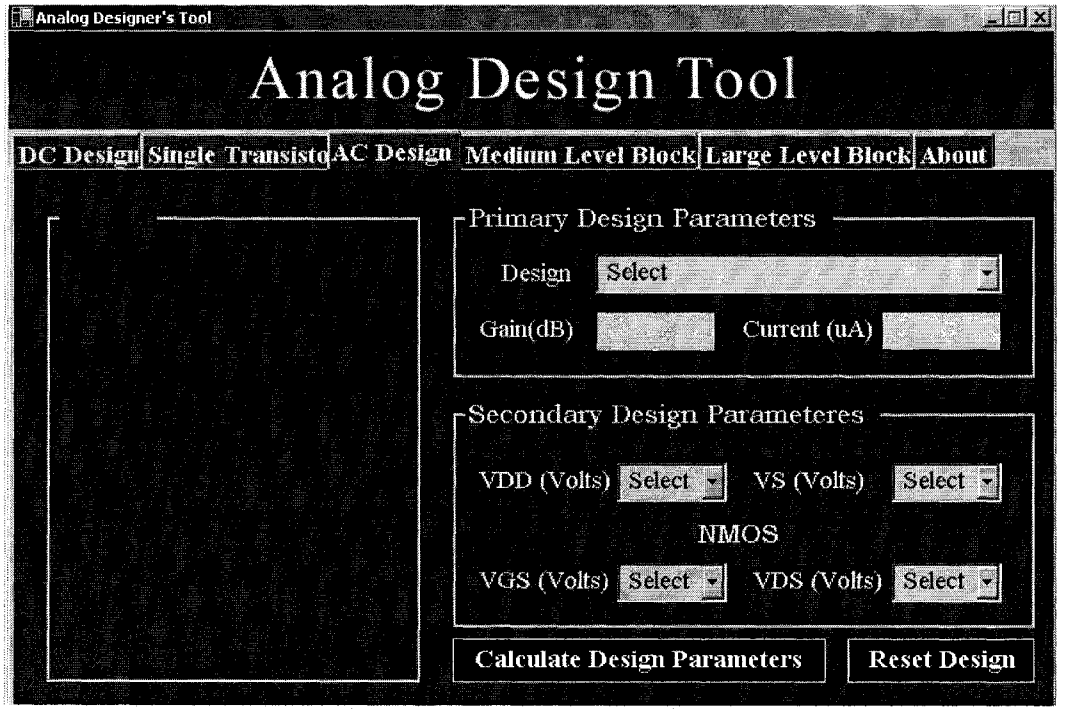


Figure C.7. Tool reset to design another problem.

APPENDIX-D

Source code for Common Source Amplifier

```
/*************************************************************************/
/*                                                                       */
/*                              Source Code                              */
/*                                                                       */
/*              Analog Integrated Circuit Design Tool                    */
/*              Common Source Amplifier Design Module                    */
/*                                                                       */
/*           Electrical & Computer Engineering Department                */
/*                       Concordia University                            */
/*                                                                       */
/*************************************************************************/

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Data.OleDb;
using System.IO;
using System.Threading;
using System.Xml.Schema;
using System.Security.Permissions;
```

```
private System.Windows.Forms.TabControl tcDesign;
private System.Windows.Forms.TabPage tpACDesign;
private System.Windows.Forms.GroupBox gbACDesign;
private System.Windows.Forms.PictureBox pbACDesign;
private System.Windows.Forms.GroupBox gbPDM;
private System.Windows.Forms.Label lblPDesign;
private System.Windows.Forms.Label lblGain;
private System.Windows.Forms.ComboBox cbDesign;
private System.Windows.Forms.TextBox tbGain;
private System.Windows.Forms.Label lblCurrent;
private System.Windows.Forms.TextBox tbCurrent;
private System.Windows.Forms.GroupBox gbSDP;
private System.Windows.Forms.Label lblVDD;
private System.Windows.Forms.ComboBox cbVDD;
private System.Windows.Forms.Label lblNMOS;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.ComboBox cbVGS;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Button cmdCDP;
private System.Windows.Forms.Button cmdRD;
private System.Windows.Forms.Label lblVS;
private System.Windows.Forms.DataGrid dgACResults;
private System.Windows.Forms.Label lblDesignName;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.ComboBox cbVS;
private System.Windows.Forms.Button cmdDCCDP;
private System.Windows.Forms.GroupBox gbDCSDP;
private System.Windows.Forms.TextBox txtVGM4;
private System.Windows.Forms.Label lblVGM4;
private System.Windows.Forms.TextBox txtVGM3;
private System.Windows.Forms.Label lblDCM3;
private System.Windows.Forms.TextBox txtVGM2;
private System.Windows.Forms.Label lblDCM2;
private System.Windows.Forms.TextBox txtVGM1;
private System.Windows.Forms.Label lblDCM1;
private System.Windows.Forms.GroupBox gbDCPDP;
private System.Windows.Forms.TextBox txtDCVB;
private System.Windows.Forms.Label lblDCVB;
private System.Windows.Forms.TextBox txtDCIDC;
private System.Windows.Forms.Label lblDCIDC;
private System.Windows.Forms.TextBox txtDCVSS;
private System.Windows.Forms.Label lblDCVSS;
private System.Windows.Forms.Label lblDCVDD;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.GroupBox gbDCDesign;
private System.Windows.Forms.PictureBox pbDCDesign;
private System.Windows.Forms.Label lblDCDesignName;
private System.Windows.Forms.TabPage tpDCDesign;
private System.Windows.Forms.Button cmdDCResetD;
private System.Windows.Forms.DataGrid dgDCResults;
private System.Windows.Forms.ComboBox cbDCVDD;
private System.Windows.Forms.ComboBox cbVDS;
private System.Windows.Forms.GroupBox groupBox2;
private System.Windows.Forms.TextBox txtTransistorC;
private System.Windows.Forms.Label lblTransistorC;
private System.Windows.Forms.ComboBox cbType;
```

```
private System.Windows.Forms.Label lblType;
private System.Windows.Forms.ComboBox cbVDSVolts;
private System.Windows.Forms.Label lblVDSVolts;
private System.Windows.Forms.ComboBox cbVGSVolts;
private System.Windows.Forms.Label lblVGSVolts;
private System.Windows.Forms.GroupBox gbTransistor;
private System.Windows.Forms.Label lblTransistorType;
private System.Windows.Forms.PictureBox pbTransistorType;
private System.Windows.Forms.GroupBox groupBox1;
private System.Windows.Forms.TextBox textBox1;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.Label label8;
private System.Windows.Forms.TextBox textBox2;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.DataGrid dataGrid1;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.ComboBox cbVSVolts;
private System.Windows.Forms.Label lblVBVolts;
private System.Windows.Forms.TextBox txtVBVolts;
private System.Windows.Forms.Button smdSTRD;
private System.Windows.Forms.Button cmdSTCDP;
private System.Windows.Forms.PictureBox pictureBox1;
private System.Windows.Forms.ImageList imglist;
private System.Windows.Forms.TabPage tpSTransistor;
private System.Windows.Forms.TabPage tpMLevel;
private System.Windows.Forms.TabPage tpLLevel;
private System.Windows.Forms.TabPage tpAbout;
private System.ComponentModel.IContainer components;


public frmAnalogTool()
{

      InitializeComponent();
      this.tcDesign.DrawMode =
                  System.Windows.Forms.TabDrawMode.OwnerDrawFixed;

      this.tcDesign.DrawItem +=
                  new DrawItemEventHandler(this.tcDesign_DrawItem);

}


private void tcDesign_DrawItem(object sender,
                              System.Windows.Forms.DrawItemEventArgs e)
{
      Font fntTab;
      Brush bshBack;
      Brush bshFore;
      fntTab = new Font(e.Font, FontStyle.Bold);

      bshBack = new
            System.Drawing.Drawing2D.LinearGradientBrush(e.Bounds,
            Color.Brown,Color.Brown,System.Drawing.Drawing2D.LinearGrad
            ientMode.BackwardDiagonal);

      bshFore = Brushes.Cornsilk;
```

```
        string tabName  = this.tcDesign.TabPages[e.Index].Text;
        StringFormat sftTab = new StringFormat();
        Rectangle  recTab = e.Bounds;
        recTab = new Rectangle(recTab.X, recTab.Y + tcDesign.Padding.Y,
                    recTab.Width ,  recTab.Height);

        e.Graphics.FillRectangle(bshBack, recTab);

        recTab = new Rectangle(recTab.X, recTab.Y + tcDesign.Padding.Y,
                    recTab.Width +500,  recTab.Height);

        e.Graphics.DrawString(tabName, fntTab, bshFore, recTab, sftTab);
}
public void Calculate_CS()
{
        int i,j;
        double Current_LL, Current_UL;
        DataRow [] PMOS;

        //Calculating NMOS Dimensions.
        ConnectToAccess("NMOS");

        //Getteing the data according to the query.
        if(Vds > 0.0) //When VDS is specified.
        {
                dtN = GetDatatable("SELECT VGS, L, WoverL, VDS, [Current],
                Gm, Gds,Gmb FROM N" + GetTableName((int)Math.Round(Vbs * -
                10.0)) + " WHERE [Current] <= " + current + "AND VGS =" +
                Vgs + " AND VDS =" + Vds);

                //Finding PMOS Match
                ConnectToAccess("PMOS");

                //Getteing the data according to the query.
                dtP = GetDatatable("SELECT VSG, L, WoverL, VSD, [Current],
                Gm, Gds,Gmb FROM P" + GetTableName((int)Math.Round(Vbs * -
                10.0))+ " WHERE [Current] <= " + current + " AND VSD =" +
                (2*Vdd - Vds));
        }
        else // When VDS is not specified
        {
                dtN = GetDatatable("SELECT VGS, L, WoverL, VDS, [Current],
                Gm, Gds,Gmb FROM N" + GetTableName((int)Math.Round(Vbs * -
                10.0))+ " WHERE [Current] <= " + current + "AND VGS =" +
                Vgs);

                //Finding PMOS Match
                ConnectToAccess("PMOS");

                //Getteing the data according to the query.
                dtP = GetDatatable("SELECT VSG, L, WoverL, VSD, [Current],
                Gm, Gds,Gmb FROM P" + GetTableName((int)Math.Round(Vbs * -
                10.0))+ " WHERE [Current] <= " + current);
        }

        //Constructing the results Table
```

```
        Construct_Results_Table();

        //Meeting the Gain Criterion
        for(i=0;i<=dtN.Rows.Count-1;i++)
        {
           Current_UL = Double.Parse(dtN.Rows[i]["Current"].ToString());
              Current_LL = Current_UL - (Current_UL * 0.01);
              LN = Double.Parse(dtN.Rows[i]["L"].ToString()+ "e-6");
              WN = LN * Double.Parse(dtN.Rows[i]["WoverL"].ToString());
            Gmn = Double.Parse(dtN.Rows[i]["Gm"].ToString());
           Gdsn = Double.Parse(dtN.Rows[i]["Gds"].ToString());
            Vds = Double.Parse(dtN.Rows[i]["VDS"].ToString());


           PMOS = dtP.Select("[Current] >= " + Current_LL + " AND
                     [Current] <= " + Current_UL);

           if (PMOS.Length >=1)
           {
              for(j=0;j<=PMOS.Length-1;j++)
              {
                    gain_obtained = Calculate_Gain(Double.Parse (PMOS[j]
                                       ["Gds"].ToString()));

                    if(gain_obtained >= gain)
                    {
                       Vsd = Double.Parse(PMOS[j]["VSD"].ToString());
                       if(Vsd + Vds == 2*Vdd)
                       {
                          Vsg = Double.Parse(PMOS[j]["VSG"].ToString());
                          LP=Double.Parse(PMOS[j]["L"].ToString()+"e-6");
                          WP = LP * Double.Parse(PMOS[j]["WoverL"]
                                .ToString ());
                          dtResults.Rows.Add(new object[] {LN, WN, Vgs,
                          Vds, LP, WP, Vsg, Vsd, Current_UL,
                          gain_obtained});
                       }
                    }
              }
           }
        }
        dtResults.AcceptChanges();
}
```