UNIVERSITY OF CINCINNATI

August 14th , 20 02

I, Avinash S. Bhangaonkar **,**
hereby submit this as part of the requirements for the degree of:

Master of Science

in:

Electrical Engineering

It is entitled:

Optimization of Performance and Sizing of Two-Stage and Folded-Cascode Op Amps

Approved by:

Dr. Joseph H. Nevin
Dr. David F. Thompson
Dr. Fred R. Beyette, Jr.

Optimization of Performance and Sizing of Two Stage and Folded Cascode Op Amps

A Thesis Submitted to the

Division of Research and Advanced Studies
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in the Department of Electrical and Computer Engineering and Computer Science
of the College of Engineering

2002

by

Avinash Bhangaonkar

B.E. University of Mumbai, 1999

Committee chair: Dr. Joseph H. Nevin

Abstract

The design of an opamp is a fairly complex task in itself. This thesis deals with the optimization of two-stage and folded cascode opamps. The optimization problems are considered to be non-linear constrained optimization problems; the constraints being non-linear. The cost function and constraints are formulated using the level 1 MOSFET model. The Matlab function 'fmincon' has been used for the actual optimization process. A GUI is provided for the user so that the specifications can be readily changed. The GUI designed in Visual Basic accepts the specifications and then writes the constraints file based on that. It then proceeds to optimize the design in Matlab. In case of a two-stage opamp, there is an option to choose the set of functions whose weight can be varied in the cost function. The cost functions are, DC gain, size, power and gain bandwidth. The optimization process creates an array of points for the design and allows the user to select any point for simulation with PSpice. For a folded cascode operational amplifier optimization, the initial guess is very important, because the problem is highly non-linear. Taking this into consideration, the cost function is kept fixed as the size and the initial guesses are varied over a wide range, which can also be changed by the user. The plots in this case are the parametrical functions of the opamp, viz. DC gain vs. size vs. power, and gain bandwidth vs. phase margin vs. slew rate. Again, the user can select any point from the plot to simulate in SPICE. The software writes a SPICE netlist for the specified point and also launches PSpice, so that the design can be readily verified. This tool has been developed for analog designers who can appreciate the tradeoffs in the design procedure. A variety of optimal points is generated so that the designer can decide for

herself/himself the specifications that can be compromised. An Excel file is written so that the details of the design elements are accessible to the user.

To my mother

Acknowledgements

First and foremost, I thank Dr. Nevin for all his guidance, support and suggestions. This thesis was an impossibility, but for the constant encouragement and a sense of direction that he provided. I am also thankful to Dr. Thompson for his kind help, time and the insights that he offered on optimization. I take this opportunity to thank Dr. Beyette for sparing time from his busy schedule to be on the defense committee.

I would like to thank my parents for creating such an environment that following this course looked so natural. The emotional support provided by my family and friends alike, was indispensable for the impetus needed to complete this successfully.

I also would like to thank all the University Libraries for their help and resources. This thesis is practically based on their hard work. The help provided by the Mathworks website has been instrumental in achieving the objective.

Table of contents

List of Figures

List of symbols

Quantity	Symbol
Design Variables	**x**
Cost Function	**f(x)**
Equality Constraints	**h(x)**
Inequality Constraints	**g(x)**
Hessian Matrix	**H**
Optimal Point	**x***
Lagrangian function	**L**
Surface mobility of the channel	μ_o
Capacitance per unit area of gate oxide	C_{OX}
Effective Gate Width	**W**
Effective Gate Length	**L**
Channel length modulation parameter	λ
Threshold voltage	V_T
Bulk threshold parameter	γ
Strong inversion surface potential	ϕ_F
Flatband Voltage	V_{FB}
Oxide charge	Q_{SS}
Boltzmann's constant	**K**
Temperature	**T**

Quantity	Symbol
Intrinsic carrier concentration	$\mathbf{n_i}$
Drain current	$\mathbf{i_D}$
Transconductance parameter	**ß**
Gate source voltage	$\mathbf{v}_{GS}$
Bulk – Source transconductance	$\mathbf{g}_{bs}$
Bulk – Drain transconductance	$\mathbf{g}_{bd}$
Channel Transconductance	$\mathbf{g}_{m}$
Drain – Source transconductance	$\mathbf{g}_{ds}$ or $\mathbf{g}_{o}$
Transfer functions	**TF**
Poles	$\mathbf{p}_1, \mathbf{p}_2 \ldots$
Zeros	$\mathbf{z}_1, \mathbf{z}_2 \ldots$
Slew Rate	**SR**
Gain Bandwidth	**GB or GBW**
Phase Margin	**PM**
Common Mode Range	**CMR**
Input voltage	$\mathbf{V}_{in}$
Output voltage	$\mathbf{V}_{out}$
Positive supply voltage	$\mathbf{V}_{DD}$ or $\mathbf{V}_{CC}$
Negative supply voltage	$\mathbf{V}_{SS}$
Coupling capacitor	$\mathbf{C}_{C}$
Load resistor	$\mathbf{R}_{L}$

Quantity	Symbol
Load capacitor	$\mathbf{C_L}$
Differential gain stage bias current	$\mathbf{I_{SS}}$
Cascode stage current	$\mathbf{I_{CASC}}$

1. Introduction

a. Need for Analog Design Automation

ASICs usually comprise of both analog and digital parts. Though the analog part occupies a very small percentage of the circuit and its area, it takes up most of the time for development. The analog part is indispensable for such circuits, because the real world inputs are invariably continuous in nature. So at the least, an A-to-D converter and D-to-A converter are integral parts of the interface of these circuits. The time-to-market and product life cycles have been reducing over the years and place additional constraints on the design development and implementation time for them. The design complexity has also increased manifold for such circuits, and only advanced CAD tools can deal with these stringent constraints. Such tools are very well developed for the digital part, but they still have a very long way to go for the analog part.

b. Current Approaches

Analog design is made up of two distinct phases, synthesis and layout[1]. Synthesis is the front end step in which the netlist is developed on the basis of the specifications for the circuit. Layout primarily concerns itself with the translation of netlists to masks.
It has been approached in two ways

1) Top Down Approach
    - Topology Selection
    - Circuit Sizing
    - Design Verification

2) Bottom Up Approach

- Layout Generation
- Detailed Design Verification

Synthesis

Synthesis has two well defined steps for its implementation, topology selection and sizing. Topology can be defined hierarchically in terms of the lower-level sub-blocks. The one that best suits the purpose is selected by the designer. The performance criteria trickle down the design path to the lowest level, so that each of the constituent blocks satisfies the required constraints. At the most fundamental level, each of these blocks is a set of devices. Each of these devices is sized according to the specifications and also taking into consideration some design objective such minimum area or minimal power dissipation[5]. Sizing essentially determines the bias parameters, element values and device sizes.

Approaches to Synthesis

Synthesis is the exact opposite of analysis, where the device sizes and bias values are given and performance is measured by using a simulator such as SPICE. Thus during synthesis the devices and bias values do not have one to one correspondence with the performance specifications. That is, for a particular set of specifications, there can be many circuit designs.

Knowledge based approach requires encoding of specific heuristic design equations in a computer executable form for the topology under consideration[8]. Such methods have an advantage in terms of fast execution speed which allows more size possibilities to be explored. But these need manual input of design equations in the form of constraints and objective function for the specified performance objectives. Thus the lead time into the project is lot higher than other symbolic simulator using methodologies. This approach keeps the door closed for automatic inclusion of new circuit schematics.

In equation based optimization approaches, analytic closed form design equations need to be hand coded to describe the system performance. In OPASYN[23] and CADICS[22] the design equations required the designer to derive and code the equations, which OPTIMAN[16] tried to solve by adding a symbolic simulator ISAAC[17] to make addition of new schematics an easier job.

An alternative approach was to simulate the entire circuit during individual iterations, so that the time needed to design the equations and code them is saved[13]. But the disadvantage of tackling the synthesis problem with such an approach is that the search space is too large leading to longer run times and it is almost impossible to guarantee a good starting point.

An in-between approach is used by ASTRX/OBLX[29] tool from CMU, where small-signal characteristics are simulated efficiently, whereas other equations have to be provided.

Closed form approaches obviously failed on the market place and the open system approaches have a large overhead of CPU times and software resources. A delicate balance has to be achieved so that the trade-off between these two conflicting requirements is taken care of.

Layout

Layout of analog circuits is more matured than synthesis because ideas from digital layout can be directly utilized for the analog counterpart. Cell layout is concerned with the conversion of transistor level schematic to a mask, whereas system assembly deals with the placement of basic functional blocks and creation of a floorplan.

Cell Layout Strategies

Changes in circuit design often require the devices to be reshaped and reoriented. Individual devices are resized and placed, while a router interconnects them taking into consideration the parasitics and couplings that affect the circuit performance. ANAGRAM and KOAN/ANAGRAM II[9] has the optimizer in the placer itself, so that the size is optimized while placing the devices. These take into account the compatibility of wires (e.g. noisy and sensitive wires) while routing them.

Latest generation of analog cell layout tools have categorized the task into two phases, device stacking followed by stack placement. The circuit is rendered as a graph of interconnected sources and drains and clusters of devices that ought to be stacks are

identified. An optimizer that judges the right stacking and placement of each stack then places these optimal stacks.

An important issue to be handled while laying out analog cells is the sensitivity of the circuit to various parasitic couplings. Symmetric cells tend be beha ve better in case of analog circuits because the parasitics associated are also symmetric. The sensitivities are handled as constraints for the layout. Since the placement and routing steps are separate, a major problem posed is the estimation of space to be left around the devices for wires. One strategy is to leave some extra space around the devices as they are placed, route them and then minimize the area by analog compaction. Another approach is to place and route simultaneously so that as the devices are placed, the wires are placed too.

The problem is further accentuated for a mixed-signal circuit. In this case the interference between the analog and the digital parts have also to be taken into consideration. The wires carrying analog signals have to be kept as far as possible from the digital lines. Also the power supply lines for these two have to be kept separate. In high performance chips, fast switching digital parts have to be removed far from the sensitive analog parts.

Thus the analog and mixed signal design is much more complex than a purely digital circuit and needs more elegant handling and rigorously built tools.

2. Symbolic Analysis

a. Definition

Symbolic analysis is a method in which closed form analytic expressions are used as opposed to the numerical analysis where the variables are represented as numbers[15]. The variables such as currents and voltages in a circuit are dependent variables, whereas time and frequency are independent variables. The behavior of the circuit is described in terms of the dependent and independent variables with the circuit elements represented as symbols.

Symbolic analysis does not actually compete with the numerical counterpart. It goes hand-in-hand with the numerical analysis in designing the circuit. Numerical analysis provides a series of numbers in a tabulated format and can verify the circuit behavior very quickly. But they are for a particular set of numbers, and can in no way suggest what parameters need to be changed to achieve the required performance. For studying the effect of changing the parameter values many simulations need to be carried out[31]. A symbolic simulator can return a first time correct analytical expressions for complex characteristics that cannot be worked out by hand. Good insight is offered by such expressions that describe the circuit behavior. The limitation of symbolic simulation is that it takes much more CPU time than numerical simulation.

b. Advantages

When using symbolic simulators for circuit sizing, the expressions need to be accurate and compact. A nested format suits such an application[11],[14]. For the analysis of circuit behavior, the expressions should rather be expanded and simplified, so that the effect of the parameters can be studied better. New circuits can be analyzed faster and more accurately with symbolic simulators[7]. The effect of adding or removing components can be immediately seen with the change in the expression. Once a code is compiled, repetitive simulations can be carried out much faster than can be for a full numerical simulation at each step.

Limitations

Analysis of circuits was restricted to lumped, linear analog circuits for a while. MOS and bipolar devices are linearized around the operating point to get the symbolic expressions for CMRR and other attributes. The expressions for analog circuits are usually very large and unwieldy. To use them for gaining insight into the circuit behavior requires them to be approximated, so that the insignificant terms in the expression are neglected. The number of terms increases exponentially with the complexity of an analog circuit. This causes a rapid increase in the memory storage and processing time, especially with the size of the circuit. One way that has been used to overcome this limitation is by using nested formula evaluation.

c. Algorithmic Aspects

The methods that have been reported in the literature of symbolic analysis use one of the following techniques

1) Matrix and determinant methods

2) Signal-flow-graph methods
3) Tree-enumeration methods
4) Parameter extraction methods
5) Interpolation methods

Signal flow graph method has been used frequently and is quite efficient. It is based on finding paths and all loops in a graph representing the circuit equations and constructed according to well-defined rules. Determinant methods are also popular and use determinants to solve the circuit equations.

Approximation of terms is also an important and involved task in itself. The coefficients of the transfer function in terms of the frequency need to be dealt with in nested expressions. So an inherent possibility is that a term that may appear insignificant at the first glance for the innermost nested expression, may gain importance for the overall expression. So cancellation of terms has to been very precise.

Hierarchical decomposition has been used for larger circuits so that the expression for the smallest possible sub-circuit is first developed and then the whole thing is combined bottom-up from there on. The Volterra series technique is used for weakly non-linear circuits, where the higher order response is added as a correction to the lower order response.

3. Optimization

a. Introduction

Improvement in computational power has been instrumental in designing complex engineering systems. Large amounts of data can be processed efficiently with the increase in computing resources. This has aided the corresponding development of designing techniques. These days solving a problem is not restricted to just getting the solution to a set of equations. The push is towards getting the best solution for the question at hand. The process of getting the best answer for a problem is optimization. As shown in Fig. 1(a) conventional design processes require the designer to use her/his experience and intuition for making the decisions. This is an advantage because conceptual changes can be brought about in the design. The conventional procedure is therefore less formal. The optimization procedure is mathematically rigorous and uses trend information to make decisions as illustrated in Fig. 1(b). The best approach for design of systems would be to integrate both the systems, so that the optimization can benefit from the designers experience[3]. At the same time, the computer conveniently handles the mathematically complex part.

b. Approaches in Optimization

Optimization is a very wide topic to be addressed completely. The classification is possible in many different ways. One could be linear and non linear optimization. When the equations for a particular problem are linear, the optimization to reach any particular objective is called linear optimization. Non-linear optimization is the one in which at least

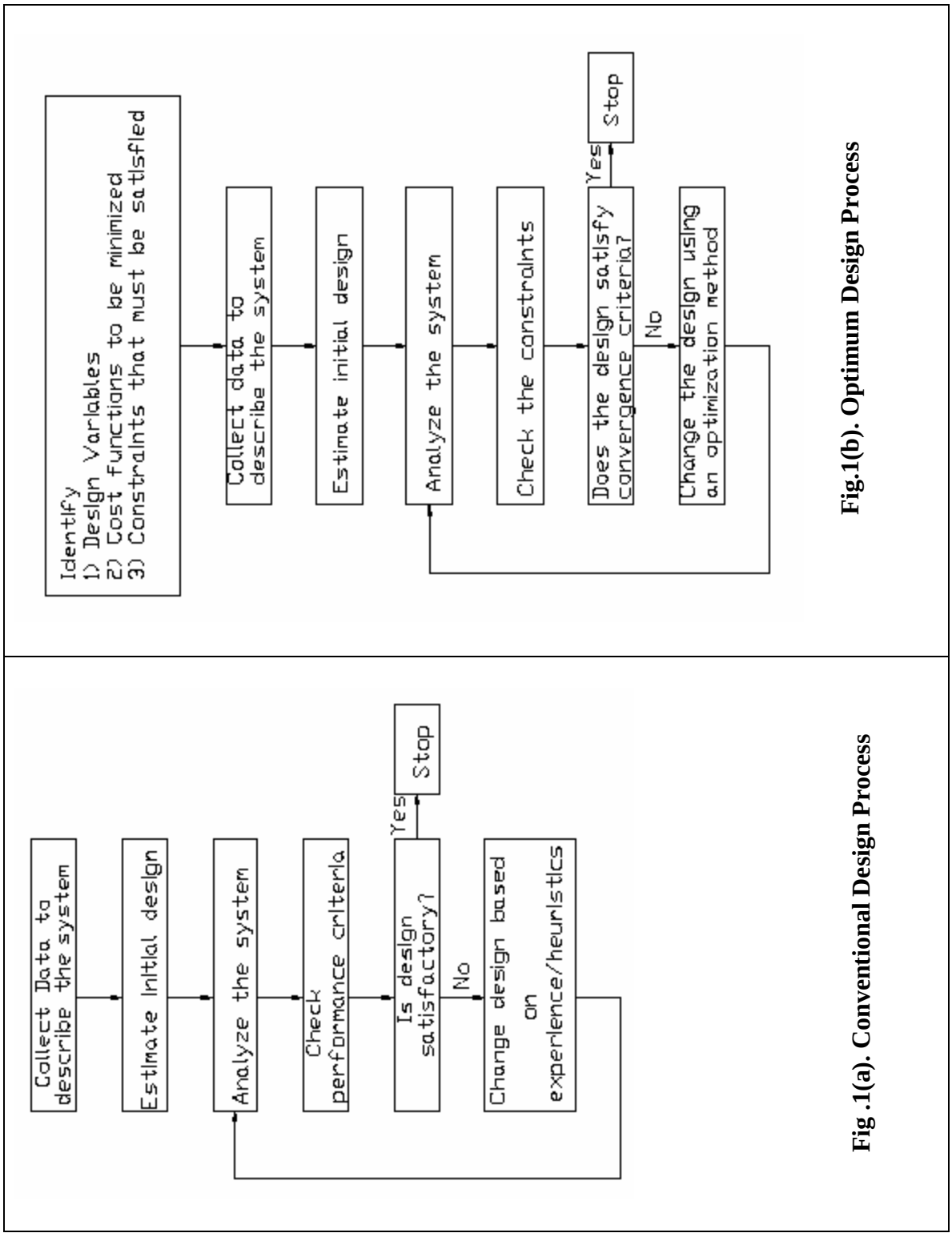


Fig.1(b). Optimum Design Process

Fig .1(a). Conventional Design Process

Note: Flow chart of the optimization processes as shown in "Introduction to Optimum Design", by Arora, J. (1989) New York: McGraw-Hill.

one of the equations is non-linear. Another classification is constrained and unconstrained optimization. When for a given problem, there are certain equations or functions that have an upper or lower bound, they effectively constrain the solution space. Hence such problems are known as constrained optimization problems. Apart from the classification of optimization, there are many different algorithms that are used to tackle these problems. Examples of these would be genetic algorithms[24], geometric optimization[20][26], quadratic programming etc.

c. Terminology

Design Variables

Parameters chosen to describe the design of a system are called the design variables. The very first and important step in the formulation of an optimization problem is choosing the set of design variables. The choice of the design variables is completely left to the designer. All options need to be investigated before zeroing on a set of the design variables. It may be beneficial to have more variables than needed for added flexibility to the problem. A numerical value can be assigned to any of those so that they are eliminated from problem formulation. Usually it is better to have design variables that are independent, so that any of those can be changed without any need to change any of the others.

They are usually represented by **'x'**.

Cost Function

Final design of a problem is completely dependent on the cost function. This is a scalar whose value is determined when the design variables are specified. It is basically a function of the design variables. Optimization is defined as the process of finding the minimum value for the cost function. For a particular problem the cost function may be a combination of many different functions. Such a problem is defined as a multi-objective design optimization[35]. There is no fixed method for solving such problems. One approach is to give weight factors to each of the components of this combined cost function. This helps in identifying the most important, because the one with a higher weight factor has a greater influence on the design. The weight factors are also needed to normalize the functions since they may not have the same order or units.

It is usually represented by f(**x**).

Design Constraints

All restrictions placed on a design are collectively called constraints. Most of the designs require certain functions to be greater or lesser than a particular value. Some variables also have conditions imposed on them. For instance, they may not make sense if negative. These conditions are translated into problem as constraints, which need to be satisfied by the design solution.

Feasibility

The problem solution that satisfies all constraints is called a feasible solution. The optimum solution obviously is a feasible solution. A solution that violates at least one constraint is called an infeasible solution.

Implicit Constraints

Some constraints are indirectly influenced by the design variables. It is not possible to express them explicitly in terms of the variables. Such constraints are called implicit constraints.

Linear/Non Linear Constraints

Constraint functions that have only first order terms of the design variables are called linear constraints. If the constraint has at least one term having order higher than one, then it is called a non-linear constraint.

Equality/Inequality Constraints

Constraints that need to have an exact value for a problem are called equality constraints. Most of the constraints need to have a value of "at least" or "at the most" a particular number. Such constraints are inequality constraints, and problems having only such constraints have inherently a bigger solution set.

Equality constraints are usually represented by $h(\mathbf{x})=0$ and inequality constraints by $g(\mathbf{x})\leq 0$.

Constraint Set

A set of all the feasible designs is known as the constraint set. Typically, it increases when the number of constraints on a problem is reduced.

Active/Inactive/Violated Constraints

For an inequality constraint, when the constraint is satisfied at the design solution i.e. $g_i(\mathbf{x}) = 0$, then it is called an active or tight or binding constraint. When the constraint is satisfied, i.e. $g(\mathbf{x}) < 0$ then it is called an inactive constraint. And for $g(\mathbf{x}) > 0$ it is called a violated constraint. Obviously, an equality constraint has to be either active or violated.

Gradient Vector

Let $f(\mathbf{x})$ be a function of n variables $x_1, x_2 \ldots x_n$. Let c_i be the partial derivative of the function with respect to x_i for i =1 to n.

Hence, $c_i = \dfrac{\partial f(x)}{\partial x_i}$ i = 1 to n … **(1)**

At any given point x*, the gradient of the function is defined as the column vector of all such partial derivatives and is represented by,

$$\nabla f(x^*) = \left[\frac{\partial f(x^*)}{\partial x_1} \quad \frac{\partial f(x^*)}{\partial x_2} \quad \cdots \quad \frac{\partial f(x^*)}{\partial x_n} \right]^T \quad \ldots \textbf{(2)}$$

Geometrically, the vector is normal to the tangent plane at **x***. It represents the direction of maximum increase in the function.

Hessian Matrix

If the gradient vector is partially differentiated again with respect to each of the elements of **x**, then the n x n matrix obtained is known as the Hessian matrix. The Hessian is denoted by **H** or $\tilde{N}^2 f$. Each element of this matrix is a function in itself that has to be evaluated at **x***. Since the function is assumed to be twice continuously differentiable, the cross partial derivatives are equal and the matrix is symmetrical.

Standard Optimization Model Attributes

1) The number of independent equality constraints must be less than or at the most equal to the number of design variables. If the number of equality constraints exceeds the number of design variables then an overdetermined set of equations exists. This may mean that there are either redundant conditions or formulation of the problem is incorrect. In case the number of constraints is equal to the number of design variables then the solution cannot be optimized because the only candidate solutions are the solutions to that set of equations. In case the number of constraints is lesser than the number of design variables, the optimum solution can be found.
2) There is no restriction on the number of inequality constraints. These constraints are always represented as ' ≤ 0 '. If there is a ' $\geq$ ' constraint, then it is multiplied by –1 and represented in the standard form.
3) If no constraints are present then it is known as an unconstrained optimization problem.
4) If all the constraints and cost function are linear, then such a problem is known as a linear programming problem and is typically easy to solve.
5) Multiplication of the cost function or any constraint does not change the design, though it changes the value of the function. Similarly addition of a positive number does not change the design.

d. Global vs. Local

Global Minimum

If a function f(**x**) has an absolute minimum at a point **x*** such that,

$f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all **x** in the feasible region, … **(3)**

then it has a global minimum at **x***.

It is called a strict global minimum if the strict inequality holds.

Local Minimum

If a function holds the inequality,

$f(\mathbf{x}^*) \leq f(\mathbf{x})$, for $\|\mathbf{x}-\mathbf{x}^*\| < d, d > 0$, … **(4)**

then it is said to have a local or a relative minimum at **x***.

Weierstrass Theorem for existence of global minimum

If f(**x**) is continuous on the feasible set S which is closed and bounded, then f(**x**) has a global minimum in S.

A closed set is the one in which the boundary points are also included and every sequence of points has a subsequence that converges to a point in the set. A bounded set is the one in which, for any point **x**, $\mathbf{x}^T\mathbf{x} < c$, where c is a finite number.

When the conditions of Weierstrass theorem are satisfied, it guarantees the existence of a global minimum. The converse is not true, i.e. a global minimum may still exist even if the conditions are satisfied. That is, it is a sufficient condition, not a necessary one.

Constrained Optimization

In the general model, the aim is to find the design variable vector x* to minimize the cost function, f(**x**)

subject to the equality constraints,

$h_i(\mathbf{x}) = 0$, i = 1 to p … **(5)**

and the inequality constraints,

$g_i(\mathbf{x}) \leq 0$, i = 1 to m ... **(6)**

A point of the design space satisfying the equality constraints is a regular point if the gradients at that point are linearly independent.

Each constraint has a multiplier associated with it, which is called a Lagrange multiplier. Lagrange multipliers are represented by 'v' for equality constraints and by 'u' for inequality constraints.

Kuhn Tucker Conditions

These are a set of necessary and sufficient conditions for regular points. Any regular point that is a minimum point must satisfy the K-T conditions.

Necessary conditions for a point x* to be an optimal point

Equality constraints

$h_i(\mathbf{x}^*) = 0$, i = 1 to p … **(7)**

$\tilde{N} f(\mathbf{x}^*) + v\tilde{N} h(\mathbf{x}^*) = 0$ … **(8)**

In words, it means that at the candidate minimum point, the gradients of cost function and constraints lie along the same line and the Lagrange multiplier is the proportionality factor. The multiplier can be both positive and negative for equality constraints. Any change in the design vector is accompanied by an increase in the cost function value when moving along a feasible direction. Any reduction in the cost function value is possible only with the violation of a constraint.

Inequality constraints

$$\nabla f(\mathbf{x}^*) + u\nabla g(\mathbf{x}^*) = 0 \text{ ... } \mathbf{(9)}$$

$$g_i(\mathbf{x}^*) \leq 0, \text{ i = 1 to m ... } \mathbf{(10)}$$

$$u_i^* \geq 0, \text{ i = 1 to m ... } \mathbf{(11)}$$

$$u_i^* g_i(\mathbf{x}^*) = 0, \text{ i = 1 to m ... } \mathbf{(12)}$$

For a problem having both equality and inequality constraints, a Lagrangian or a Lagrange function can be defined as

$\mathbf{L} = f(\mathbf{x}) + \mathbf{v}^T\mathbf{h}(\mathbf{x}) + \mathbf{u}^T\mathbf{g}(\mathbf{x}) \ldots$ **(13)**

And the necessary condition becomes, $\nabla\mathbf{L} = 0 \ldots$ **(14)**

in addition to the constraint satisfaction and the requirement that the Lagrange multipliers of the inequality constraints be non-negative.

e. Convex vs. Non-Convex

A question of global optimality can be tackled in two different ways.

1) For a closed and bounded set, calculate all the optimum points, and the one having the least function value is the global optimum. The Weierstrass theorem guarantees global optimality in such a case.
2) Show that the optimization problem is convex, so that the local optimum is in fact the global one.

Convex Sets

If P_1 and P_2 are any points in a convex set S, then the entire line segment P_1-P_2 is also in S.

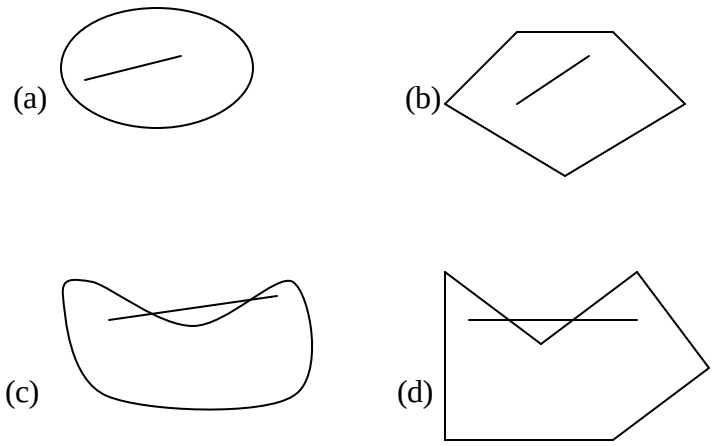


Fig.2. (a) and (b) Convex Sets (c) and (d) Non-Convex Sets

For an n dimensional space, the line segment between any points x(1) and x(2) can be written as,

$\mathbf{x} = \alpha\, \mathbf{x}^{(1)} + (1-\alpha)\, \mathbf{x}^{(2)}; 0 \le \alpha \le 1$ … **(15)**

Convex Functions

A function is said to be convex if it lies below the line joining any two points on the curve f(**x**).

For an n dimensional space this can be mathematically represented by

$f(\alpha\, \mathbf{x}^{(2)} + (1-\alpha)\, \mathbf{x}^{(1)}) \leq \alpha\, f(\mathbf{x}^{(2)}) + (1-\alpha)\, f(\mathbf{x}^{(1)})$; for $0 \leq \alpha \leq 1$ ... **(16)**

for any two points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ in S.

This can be difficult to check since it requires taking into consideration an infinite number of pairs of points. An easier method to verify the convexity of a function is by evaluating the Hessian of the function. If the Hessian of a function is positive semidefinite, or positive definite at all points, then the function is convex. If the Hessian is positive definite, then the function is strictly convex.

Convex Programming Problem

The intersection of convex sets is always convex. For all convex $g_i(\mathbf{x})$, $g_i(\mathbf{x}) \leq e_i$ is also convex, where e_i is a constant. Intersection of all $g_i(\mathbf{x}) \leq e_i$ gives rise to a convex set.

The set S defined by

$S = \{\mathbf{x} \mid g_i(\mathbf{x}) \leq 0,\ i = 1 \text{ to } m\ ;\ h_j(\mathbf{x}) = 0,\ j = 1 \text{ to } p\}$ … **(17)**

is convex if g_i are convex and h_j are linear.

If any of the equality constraints is non-linear, then the set S is always non-convex. A constraint has to be convex if it has only linear equality and inequality constraints.

If for a particular problem, the set S is convex and the cost function is also convex over the set S, then it is a convex programming problem.

For any Convex Programming Problem, the local minimum is also the global minimum.

However convexity is a sufficient condition for global optimality. A non convex problem can have a global optimum, but to claim that requires a more exhaustive search of the design variable space.

The second order necessary and sufficient conditions for an optimal point are obtained by writing the Taylor expansion of the Lagrange function in the feasible direction **d**.

The necessary condition for the point to be a minimum point is that the second order term of the expansion must be non-negative. For the necessary conditions all active inequalities with non-negative multipliers are considered for determining the direction **d**.

The sufficient condition is that the second order term is positive for all **d** in the constraint tangent plane. For the sufficient condition only the active inequalities with positive multipliers are considered for the determination of direction **d**.

According to the strong sufficient condition, if $\tilde{N}^2 L(\mathbf{x}^*)$ is a positive definite matrix, then the point is an isolated minimum point.

f. Optimization in Matlab

For this work, Matlab was used as a tool for optimization. The 'fmincon' function was used in particular. This is how the fmincon algorithm works.

In constrained optimization, the general aim is to transform the problem into an easier subproblem that can then be solved and used as the basis of an iterative process.

The solution of the KT equations forms the basis to many nonlinear programming algorithms. These algorithms attempt to compute directly the Lagrange multipliers. Constrained quasi-Newton methods guarantee superlinear convergence by accumulating second order information regarding the KT equations using a quasi-Newton updating procedure. Matlab uses these Sequential Quadratic Programming methods, which solve a Quadratic Programming Problem at every major iteration.

At each major iteration an approximation is made of the Hessian of the Lagrangian function using a quasi-Newton updating method. This is then used to generate a QP subproblem whose solution is used to form a search direction for a line search procedure.

An SQP Algorithm has the following major steps

1) Hessian Matrix update of the Lagrangian function
2) QP Problem Solution
3) Line search and Merit function calculation

Updating the Hessian Matrix

At each major iteration a positive definite quasi-Newton approximation of the Hessian of the Lagrangian function, *H*, is calculated using the BFGS method. When the Hessian has to be modified using the first phase of the procedure to keep it positive definite, then "Hessian modified" is displayed. If the Hessian has to be modified again using the second phase, then "Hessian modified twice" is displayed. When the QP subproblem is infeasible, then "Infeasible" is displayed. Such displays are usually not a cause for concern but indicate that the problem is highly nonlinear and that convergence may take longer than usual. Sometimes the message "No update" is displayed. This can be an indication that the problem setup is wrong or that the function is discontinuous.

QP Problem Solution

At each major iteration of the SQP method a QP problem is solved. The method used in the Optimization Toolbox is an active set strategy. It has been modified for both Linear Programming (LP) and Quadratic Programming (QP) problems.

The solution procedure involves two phases: the first phase involves the calculation of a feasible point (if one exists), the second phase involves the generation of an iterative sequence of feasible points that converge to the solution. In this method an active set is maintained, which is an estimate of the active constraints (i.e., which are on the constraint boundaries) at the solution point.

The active set is updated at each iteration, and this is used to form a basis for a search direction. Equality constraints always remain in the active set. The search direction is calculated that minimizes the objective function while remaining on any active constraint

boundaries. The search direction is guaranteed to remain on the boundaries of the active constraints.

4. Algorithm Development

Many designers using a multitude of techniques have approached analog circuit design automation. There are always some advantages and some limitations for any particular method. So choosing a method for analog synthesis is somewhat “analog” in itself. There are trade-offs to handle when a method is decided upon. The most common trade-off is that between CPU time and accuracy. As the equations become more and more accurate, their complexity naturally increases and so does the burden on the CPU.

a. Intended Audience

This work is concentrated on analog synthesis. It basically aims to build a frontend tool that will convert a set of specifications to a sized circuit. The backend, i.e. the conversion from circuit to a layout is not in the scope of this work. This has been designed primarily for a designer who has at least some understanding and appreciation of analog circuit design. Opamps are the building blocks of many analog circuits and hence have been dealt in detail in this work. Designing an opamp is a very involved task in itself. So to build up circuits containing one or more opamps is a challenge even for an expert designer. This work tries to ease that aspect of the circuit design. If the technique needed to synthesize the device sizes from circuit specifications of an opamp is made available to a designer, then she/he can concentrate on other areas of the design.

b. Targeted Problem

Using automated design helps further in optimizing the sizes of the devices for a particular cost function. Usually one of the parameters such as power dissipation, or area or DC gain is considered as a cost function and others are constraints for the design. Expert designers use their experience and rules of thumb to size the circuit. This does not push the circuit design to the edge, and hence is more conservative in nature. The circuit design usually stops when the designer is satisfied that it will work and is within reasonable limits. Thus seldom does such a hand-designed circuit give the very best performance it can. The aim of this work is to relieve the designer of doing the iterations needed to get the circuit designed.

Nonlinear circuits exhibit a very subtle quality of being sensitive to initial guess. Many consider this as a problem and thus they try to linearize the equations as much as possible to get the same solution for any starting point. On the contrary this nonlinearity provides richness to the designing alternatives. A small change in the initial point can give rise to a dynamic difference in the solution. This behavior can be exploited to the fullest to generate a larger solution space. Inherently such a design problem can offer more options during optimization than a linear one. Thus the nonlinearity has been left as is in the design equations. The designer can choose from the array of solutions created because of different starting guesses. The variety provided by such a handling of the problem outweighs the limitations due to the complexity of the equations and the resultant computational time.

This tool has been designed keeping in mind users who will be able to appreciate the tradeoffs between different design constraints. For example, the slew rate of an opamp decides the speed with which the opamp reacts to a change in signal. For a two-stage opamp it is defined as a ratio of the current in a particular branch to the coupling capacitor. And the capacitor value cannot be reduced below a certain limit. So the way to increase the slew rate is by increasing the current. But this implicitly means that the power dissipation associated with the circuit is increased. Thus depending on the initial guess, the program returns different solutions, some of which have a higher slew rate. These designs also show higher power dissipation. The designer using this software should be able to judge the best design to suit her/his purpose. For instance, some applications may require the design to be dissipating the least possible power, even if it is at the expense of speed of response from the circuit. On the other hand some applications might need a faster change in output with a change in input, and may be able to tolerate higher power.

c. Approach used and its justification

This tool is designed so that it can be used on as broad a gamut of machines as possible. It was imperative to use standard software to meet this end. Hence MATLAB was chosen as the workhorse for this project. MATLAB also suits the requirement because it has an built-in subroutine for constrained optimization problems. In particular ‘fmincon’ was used for the optimizer.

d. Algorithm and the flow of logic

At the heart of the algorithm are three files that interact with each other, as shown in Fig.3.

- The first one contains information regarding the cost function. It contains the cost function itself and its gradient. Though providing the gradient is not strictly necessary, it does help processing the program faster. It also has global and other local variables needed to compute the cost function value.
- The second file stores all the constraints for the current problem and its gradients. It definitely helps to provide gradients in this file as well because there are numerous constraints, so the processing time is reduced to a larger extent.
- The third file is the main file that basically does all the processing. The first two files are function files that just return the cost function, constraints and their gradients. This file accepts the function values from the previous files and uses it to generate an optimum point for the current case. The circuit equations are non-linear and hence the solution definitely depends on the initial guess. So numerous guesses are tried to generate a large solution space. This offers further flexibility to the designer.

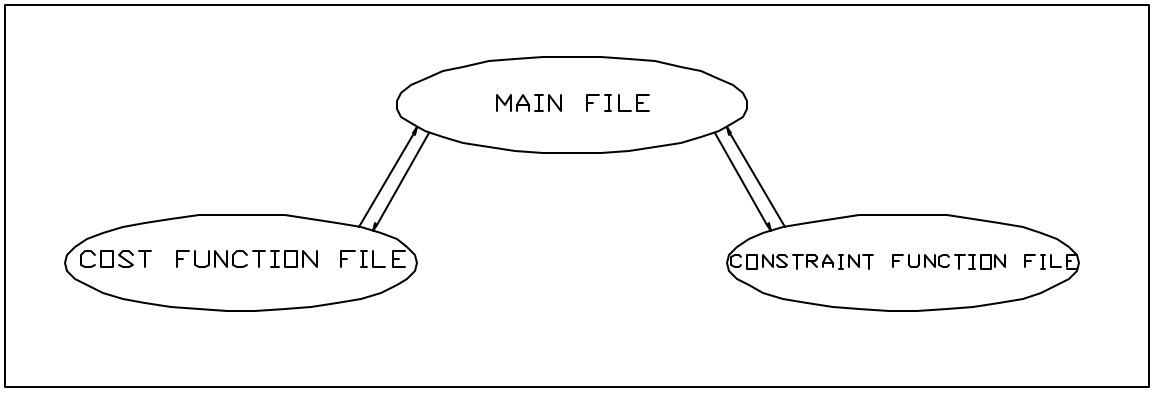


Fig.3. Interaction among MATLAB files.

Though the basic approach to the optimization is the same, there is a subtle difference between the two-stage and folded cascode optimization.

The two stage optimization problem behaves like a convex programming problem. For about 800 different starting points, the solution was found to be the same. This completely defeats the purpose of giving a large solution space to the user. This shortcoming is annulled by varying the cost function. The problem is treated as a multi-objective constrained optimization.

The obvious choices for the cost function variables are DC Gain, Power Dissipation and Size of the circuit. The first variable needs to be maximized and the other two minimized. There are two major hurdles in such multi-objective optimization. Firstly the units of these functions are not the same, so they need to be normalized before clubbing them together. This normalization is usually done by dividing the individual cost functions by

their respective values at the initial guess. The other important issue to be tackled is the magnitude of these functions. For instance, the DC gain is in thousands, power dissipation is in hundreds and size ranges from tens to hundreds, depending on the specifications. These two things are simultaneously taken care of by the cost multipliers function multipliers. The multipliers are then varied with successive iterations so that an array of solutions can be obtained.

For the folded cascode case, the cost function is kept fixed as the size of the circuit. Here just changing the initial variable values yields a variety of different solutions. They are then represented graphically in the form of plots with the axes as power dissipation, size and DC gain, and slew rate, phase margin and gain bandwidth. The designers then have the choice of selecting any of the solutions and verify the design using SPICE.

e. Interface Development

The interface for this software has been designed in Visual Basic and MATLAB[27]. There is a tradeoff between user flexibility and designing complexity for any kind of software. A designer will definitely like to have as much flexibility as possible, which in turn means additional coding overhead for the software designer. A concerted effort has been made to optimize the flexibility keeping a track of the constraint of design complexity.

In order to elucidate the interface design, a flow chart has been prepared to describe the logical trend in the program.

As seen in the flow charts in Fig.4 and 5, there is a need for an interface right at beginning, where the circuit specifications are read in. These are useful for creating the constraint equations for the problem. The specifications are maximum power dissipation, minimum slew rate, maximum and minimum common mode ranges and outputs, minimum phase margin, minimum gain bandwidth product, and minimum DC gain. In addition to reading in these values, the interface has five buttons.

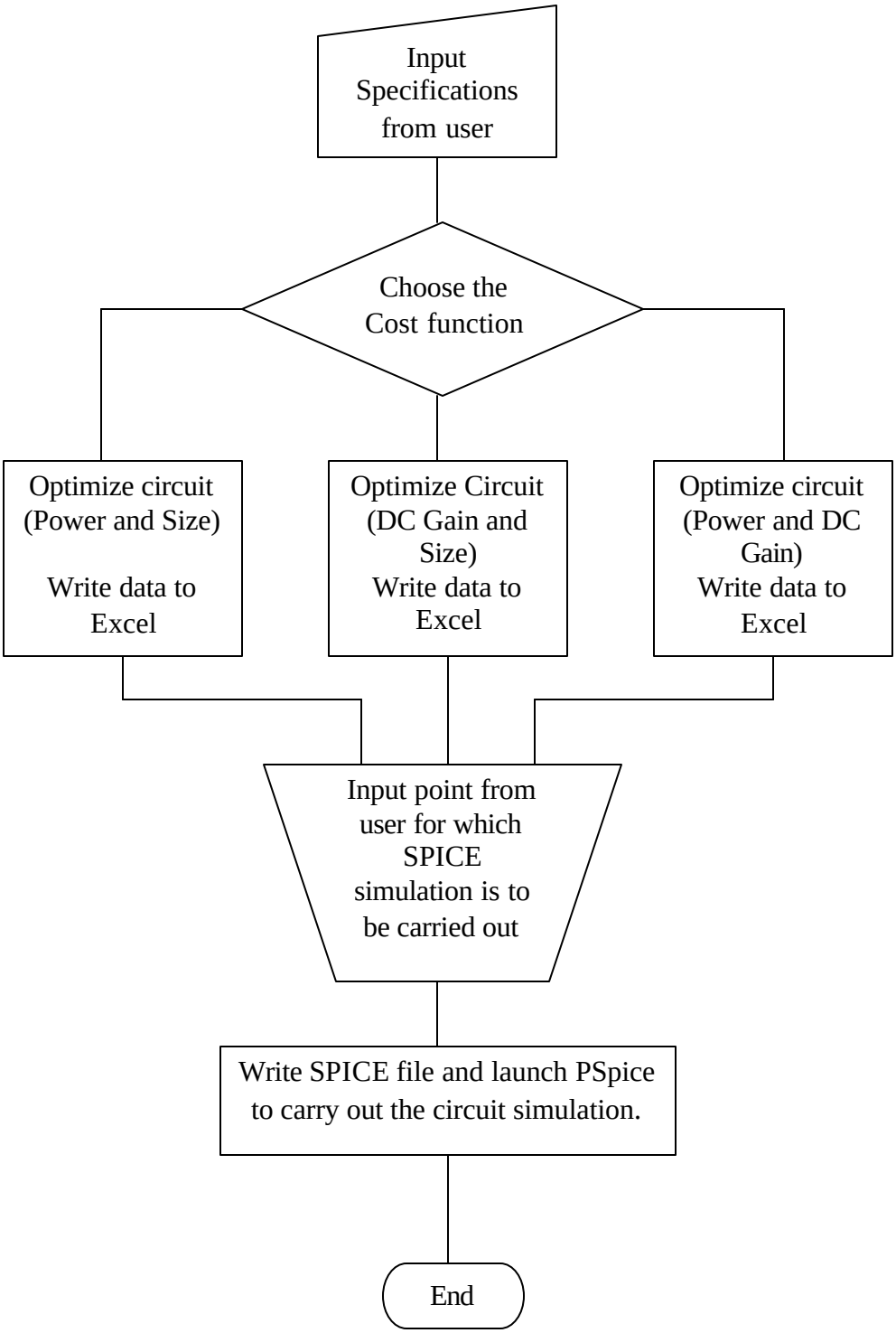


Fig.4. Flow Chart for Two Stage Op-Amp Optimization

1)

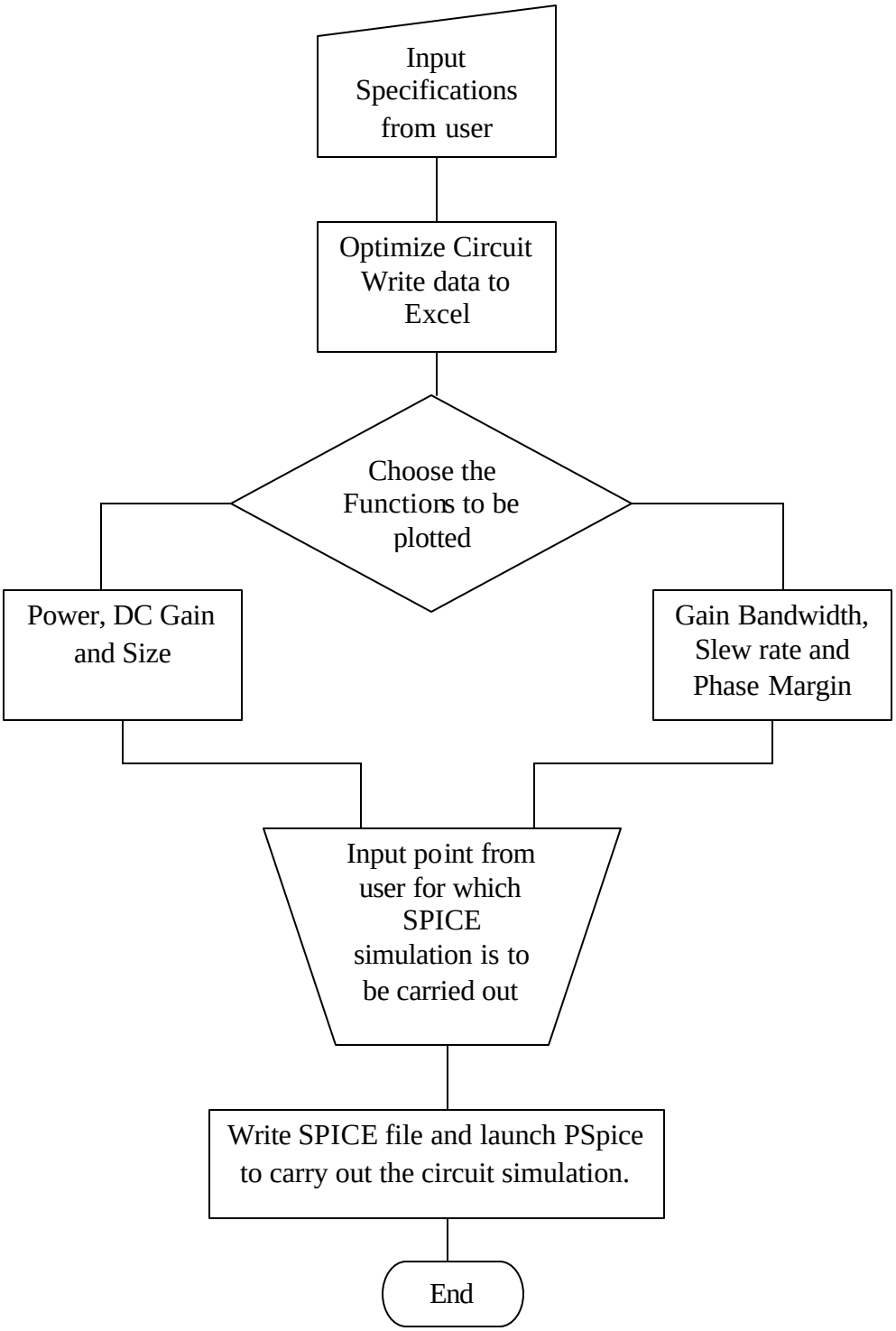


Fig.5. Flow Chart for Folded Cascode Op-Amp Optimization

The five functions that the buttons on the interface perform are as follows

Submit

This button accepts the specifications from the user. It checks whether all the fields are provided and then proceeds to write a constraint file in Matlab. If any of the field is left empty, then it displays an error message to the user. It also asks for the path where all other files are stored and sets that path for rest of the files. The constraint file is written in the same directory as the path.

Reset

This button provides default values to the specifications. It also displays a message to the user asking her/him to press the "Submit" button if the values are correct and acceptable.

Optimize

When clicked it first launches Matlab and then executes a couple of commands in the Matlab environment. It sets the path for Matlab and then executes the optimization routine, which is assumed to be stored in the same directory.

Write SPICE File

It reads the array index number written by the Matlab routine. It then reads the details for those particular solutions from the Excel file written previously during Matlab simulation. It then writes a SPICE file in the same directory.

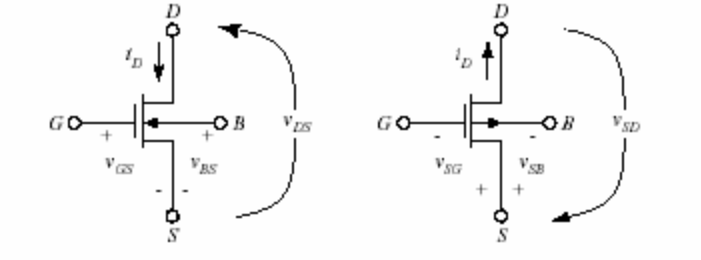
Launch PSpice

On clicking this button PSpice is launched.

5. Two-Stage OpAmp Circuit

Transistor Model

A model is a representation of a device in the form of equations so that the behavior of a circuit can be predicted or the performance of the system verified. An MOS transistor is a non-linear device because of the large signal I-V characteristic. However, it can be linearized around the operating point for the small signal model[2]. The voltage and current variables for N and P channel MOSFETs are shown in Fig. 6.



N-channel P-channel

Fig.6. Conventions and symbols for N and P channel MOSFETs

a. Large Signal Model

The Shichman and Hodges Model or Level 1 SPICE model is described here for an NMOS device[6][10][33]. The same model can be used to describe a PMOS device by multiplying all the voltages and currents by -1 and using the absolute value for p-channel threshold.

This model is developed from the following equation

$$i_D = \frac{m_0 C_{ox} W}{2L}\left[(v_{GS} - V_T) - \left(\frac{v_{DS}}{2}\right)\right] v_{DS} \quad \ldots \textbf{(1)}$$

The parameters in this equation are

μ_0 = surface mobility of the channel for the NMOS and PMOS device (cm^2/V-s)

$C_{ox} = e/t_{ox}$ = capacitance per unit area of the gate oxide (F/cm^2)

W = Effective Gate Width

L = Effective Gate Length

λ = channel length modulation parameter (volt $^{-1}$)

The threshold voltage is given by

$$V_T = V_{TO} + \left[\sqrt{2|\Phi_F| + v_{SB}} + \sqrt{2|\Phi_F|}\right] \ldots \textbf{(2)}$$

$$V_{TO} = V_T(v_{SB} = 0) = v_{FB} + 2|\Phi_F| + \frac{\sqrt{2q e_{Si} N_{SUB} 2|\Phi_F|}}{C_{ox}} \quad \ldots \textbf{(3)}$$

γ = bulk threshold parameter (V ^{½}) $= \dfrac{\sqrt{2 e_{Si} q N_{SUB}}}{C_{ox}}$ … **(4)**

Φ_F = strong inversion surface potential (V) $= -\dfrac{kT}{q} \ln\left(\dfrac{N_{SUB}}{n_i}\right)$ … **(5)**

V_{FB} = flatband voltage (V) = $\Phi_{MS} - \dfrac{Q_{SS}}{C_{ox}}$ … **(6)**

Φ_{MS} = Φ_F (substrate) - Φ_F (gate) … **(7)**

Φ_F (substrate) $= -\dfrac{kT}{q} \ln\left(\dfrac{N_{SUB}}{n_i}\right)$ [n-channel with p-substrate] … **(8)**

$$\mathbf{F}_F(\text{gate}) = -\frac{kT}{q}\ln\left(\frac{N_{GATE}}{n_i}\right)$$ [n-channel with n^+ polysilicon gate] … **(9)**

Q_{SS} = oxide charge = q N_{SS} … **(10)**

k = Boltzmann's constant

T = temperature (K)

n_i = intrinsic carrier concentration

The behavior of MOS transistors depends on the source to bulk voltage and hence they are treated as four terminal devices. The equations are better represented by electrical parameters rather than physical ones when being used for circuit design[18]. Hence the drain current is often expressed as

$$i_D = b\left[(v_{GS}-V_T)-\frac{v_{DS}}{2}\right]v_{DS} = K'\frac{W}{L}\left[(v_{GS}-V_T)-\frac{v_{DS}}{2}\right]v_{DS}$$ … **(11)**

where the transconductance parameter ß is defined in terms of physical parameters as

$$b = (K')\frac{W}{L} \cong (m_0 C_{ox})\frac{W}{L}$$ (A/V^2) … **(12)**

There are various regions of operation for an MOS transistor. These regions depend on the value of (v_{GS}-V_T).

When (v_{GS}-V_T) = 0, the MOSFET is said to be in the cutoff region and the channel acts like an open-circuit.

I_D = 0, (v_{GS}-V_T) = 0 … **(13)**

The next two regions of operation depend on the value of $v_{DS}(sat) = (v_{GS}-V_T)$.

If v_{DS} is less than $v_{DS}(sat)$ then the transistor is in the non-saturated region and the drain current is given by

$$i_D = K'\frac{W}{L}\left[(v_{GS}-V_T)-\frac{v_{DS}}{2}\right]v_{DS} \;;\; 0 < v_{DS} \le (v_{GS}-V_T) \;\ldots \textbf{(14)}$$

For v_{DS} greater than $v_{DS}(sat)$, the drain current becomes independent of v_{DS} and transistor is said to be in saturation. In such an operation, the current is given by

$$i_D = K'\frac{W}{2L}(v_{GS}-V_T)^2 \;;\; v_{DS} \ge (v_{GS}-V_T) \;\ldots \textbf{(15)}$$

In reality the drain current is not independent of the drain-source voltage because the channel length is shortened as the voltage increases, resulting in higher current. This phenomenon is called channel length modulation and is incorporated in the equation by introducing a parameter λ. The factor multiplied in the equation is $(1+\lambda v_{DS})$. And the entire equation is given as,

$$i_D = K'\frac{W}{2L}(v_{GS}-V_T)^2(1+\lambda v_{DS}); \; v_{DS} \ge (v_{GS}-V_T) \;\ldots \textbf{(16)}$$

A circuit version of this large signal model has a current source that depends on the drain, source, and bulk and gate voltages. This model is completely defined by λ, V_T, γ, $2|\phi_F|$ and K'. The function of the large signal model is to solve for the drain current and terminal voltages of the device.

The charge storage capacitances are given by following relations

Off

$C_{GB} = C_{ox} W_{eff} L_{eff} + 2 CGBO L_{eff}$ … **(17)**

$C_{GS} = C_{ox} LD W_{eff} = CGSO W_{eff}$ … **(18)**

$C_{GD} = C_{ox} LD W_{eff} = CGDO W_{eff}$ … **(19)**

Saturation

$C_{GB} = CGBO L_{eff}$ … **(20)**

$C_{GS} = C_{ox} (LD + 0.67 L_{eff}) (W_{eff}) = CGSO W_{eff} + 0.67 C_{ox} (W_{eff}) (L_{eff})$ … **(21)**

$C_{GD} = C_{ox} LD W_{eff} = CGDO W_{eff}$ … **(22)**

Non – Saturation

$C_{GB} = CGBO L_{eff}$ … **(23)**

$C_{GS} = C_{ox} (LD + 0.5 L_{eff}) (W_{eff}) = CGSO W_{eff} + 0.5 C_{ox} (W_{eff}) (L_{eff})$ … **(24)**

$C_{GD} = C_{ox} (LD + 0.5 L_{eff}) (W_{eff}) = CGDO W_{eff} + 0.5 C_{ox} (W_{eff}) (L_{eff})$ … **(25)**

The large signal model is shown in Fig. 7.

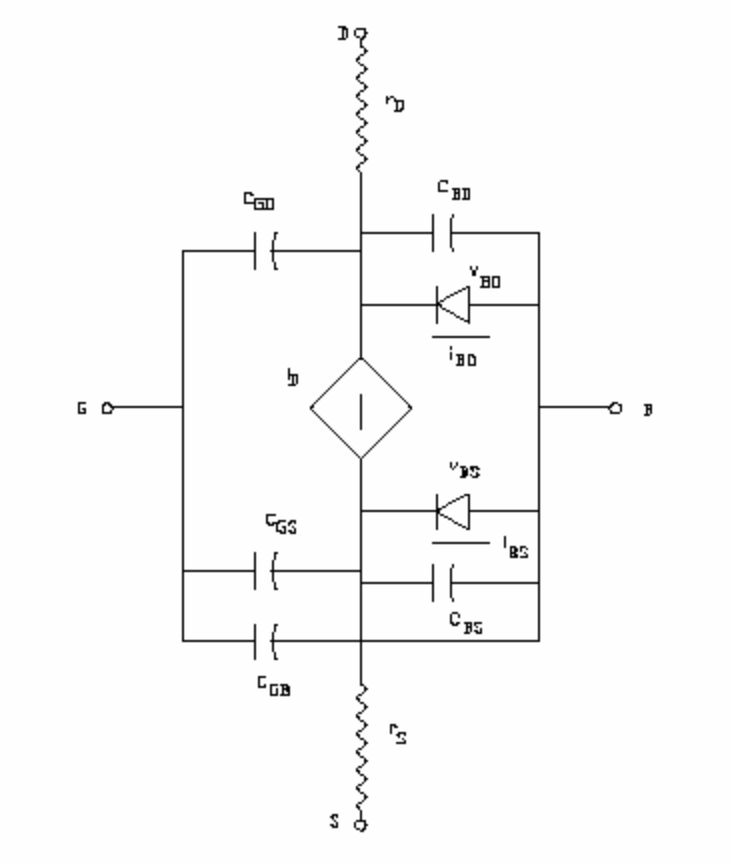


Fig.7. Large Signal Model

b. Small Signal Model

Once the operating point is determined, the small signal model comes into play. It is a linearized model that is valid only over a small range where large signal voltages and currents can be represented by a straight line.

Small signal model parameters are represented as the ratio of small perturbations of large signal models or as a partial differentiation of one large signal parameter with respect to another.

The bulk-drain and bulk-source conductances are negligible at the operating point, as these junctions are normally reverse biased.

$$g_{bd} = \frac{\partial I_{BD}}{\partial V_{BD}} \cong 0 \text{ (At the quiescent point)} \ldots \textbf{(26)}$$

$$g_{bs} = \frac{\partial I_{BS}}{\partial V_{BS}} \cong 0 \text{ (At the quiescent point)} \ldots \textbf{(27)}$$

The channel conductances are defined as

$$g_m = \frac{\partial I_D}{\partial V_{GS}} \text{ (At the quiescent point)} \ldots \textbf{(28)}$$

$$g_{mbs} = \frac{\partial I_D}{\partial V_{BS}} \text{ (At the quiescent point)} \ldots \textbf{(29)}$$

$$g_{ds} = \frac{\partial I_D}{\partial V_{DS}} \text{ (At the quiescent point)} \ldots \textbf{(30)}$$

The values that these small signal parameters take depend on the region of operation of the transistor. Thus these values are completely dependent on the large signal parameters.

The values in saturation region are given by

$$g_m = \sqrt{2K' \frac{W}{L} |I_D| (1 + l V_{DS})} \cong \sqrt{2K' \frac{W}{L} |I_D|} \ldots \textbf{(31)}$$

$$g_{mbs} = g_m \frac{g}{2\left(2|\Phi_F| + V_{SB}\right)^{1/2}} = h\, g_m \ldots \textbf{(32)}$$

$$g_{ds} = g_o = \frac{I_D l}{1 + l V_{DS}} \cong I_D l \ldots \textbf{(33)}$$

In the non-saturation region the values of the small signal parameters are given by the following equations.

$$g_m = b\,V_{DS}(1 + l\,V_{DS}) \cong b\,V_{DS} \quad \ldots \textbf{(34)}$$

$$g_{mbs} = \frac{b\,g\,V_{DS}}{2\left(2|\Phi_F| + V_{SB}\right)^{1/2}} \quad \ldots \textbf{(35)}$$

$$g_{ds} = g_o = b\left(V_{GS} - V_T - V_{DS}\right)(1 + l\,V_{DS}) + \frac{I_D\,l}{1 + l\,V_{DS}} \cong b\left(V_{GS} - V_T - V_{DS}\right) \quad \ldots \textbf{(36)}$$

Fig. 8 shows the small signal model.

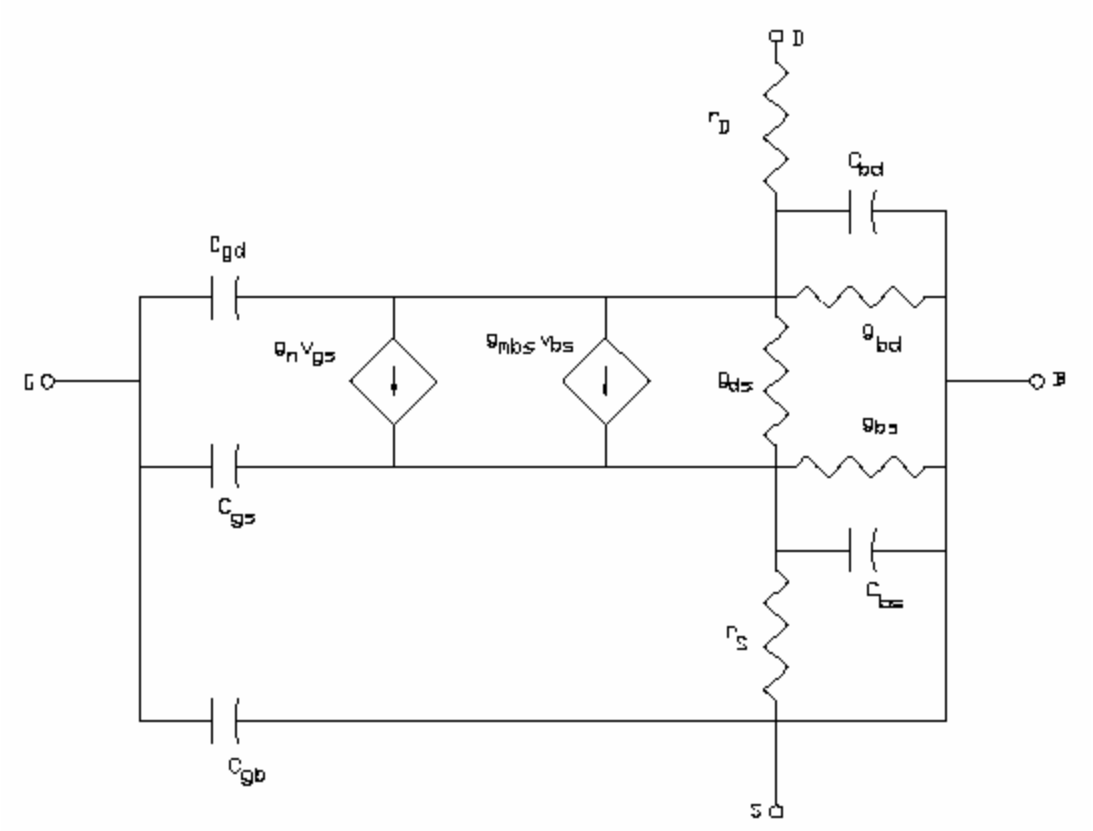


Fig.8. Small Signal Model

The large signal model used here neglects many second order effects, especially the ones due to short and narrow channel dimensions. But this model has been chosen because its simplicity allows equations to be developed for complex circuits such as the opamps.

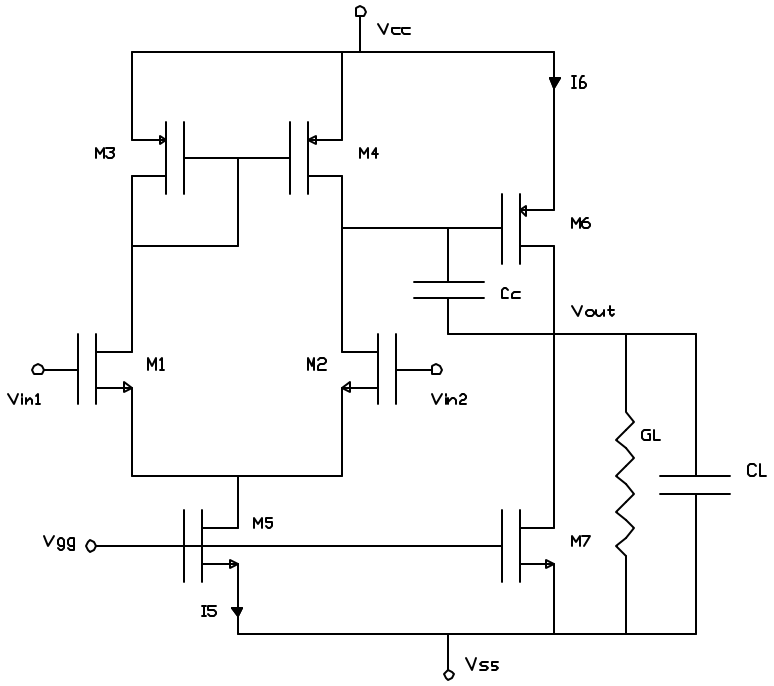


Fig.9. Two-Stage Operational Amplifier

c. Description of the circuit

Fig. 9 is a schematic of a typical two-stage opamp[19][21]. Transistors M1 and M2 are NMOS transistors and form the differential input gain stage. Gate of M1 is the inverting terminal and that of M2 is the non-inverting input of the opamp. The gain of this stage is the transconductance of M2 multiplied by the output resistance at its drain. The output resistance is contributed by the differential input transistors and the current mirror pair M3 and M4. The current mirror pair increases the resistance in a much smaller area than a passive resistance and also converts the dual ended input to single ended output. The differential input does not really contribute towards the actual gain. This is provided by the second stage formed by M6 and M7. The PMOS transistor M6 is in the common source mode and hence provides a good transconductance value. The resistance at the

drain of M6 is the output resistance of M6 itself, output resistance of M7 and the load resistance.

d. Mathematical Equations

The transfer function of the opamp in Fig.9, as returned by the program SAPWIN[34], is

$$TF = \frac{g_{m2}g_{m6} + s\,g_{m2}C_C}{s^2 C_L C_C + s\,C_C g_{m6} + (g_{ds2} + g_{ds4})(g_{ds6} + g_{ds7} + G_L)} \quad \ldots \textbf{(37)}$$

The poles and zeros of this circuit can be determined by assuming that the poles and zeros are far apart. They are

$$p_1 = -\frac{(g_{ds2} + g_{ds4})(g_{ds6} + g_{ds7} + G_L)}{C_C\, g_{m6}} \quad \ldots \textbf{(38)}$$

$$p_2 = -\frac{g_{m6}}{C_L} \quad \ldots \textbf{(39)}$$

$$z_1 = \frac{g_{m6}}{C_C} \quad \ldots \textbf{(40)}$$

The specifications for an operational amplifier circuit are given in terms of parameters such as the DC gain, unity gain bandwidth product, phase margin, slew rate and the input and output ranges. These have to be expressed in terms of the sizes of the transistors so that they may be used for the optimization procedure.

$$SR = \frac{I_5}{C_C} \quad \ldots \textbf{(41)}$$

$$GB = \frac{g_{m2}}{C_C} \quad \ldots \textbf{(42)}$$

$$V_{in(\max)} = V_{CC} - \left(\frac{I_5}{b_4}\right)^{1/2} - \left|V_{TO3}\right|_{(\max)} + V_T \quad \ldots \textbf{(43)}$$

$$V_{in(\min)} = V_{SS} + \left(\frac{I_5}{b_3}\right)^{1/2} + V_{T(\max)} + V_{DS5\,(sat)} \quad \ldots \textbf{(44)}$$

$$V_{out(\max)} = V_{CC} - V_{ds6\,(sat)} = V_{CC} - \frac{2I_6}{g_{m6}} \quad \ldots \textbf{(45)}$$

$$V_{out(\min)} = V_{SS} + V_{ds7\,(sat)} = V_{SS} + \frac{2I_7}{g_{m7}} \quad \ldots \textbf{(46)}$$

e. Optimization approach

The sizing problem is a non-linear optimization issue that has been solved by using the optimization subroutine of MATLAB. Here a cost function is provided and design specifications are considered to be constraints for the optimization problem. The non-linear constraints render the design problem to be non-convex. This leads to the issue of sensitivity of the design to the initial guess. There are two ways in which this topic is perceived. This non-linearity causes the dynamics of the optimization problem to be rich in behavior. There is a possibility of getting very good solutions in such cases. If, on the other hand, the problem is conceived to be a convex one, then the final design is independent of the starting point and for the given set of specifications there is only a single optimal solution.

The routine perceives every variable and parameter as just a number. To keep the program as accurate and as effective as possible, it is necessary to decide the units for the variables and other parameters so that they do not become unwieldy for the optimization

software. Values of capacitances are seldom more than a few nano-farads whereas the Unity Gain Bandwidth is usually in MHz. Thus in the same problem there are about 15 orders of magnitude difference. For the routine to work properly it is imperative to keep the orders of all the quantities about the same. Otherwise the variables with greater magnitudes render the smaller quantities ineffective.

For this particular problem, the units are decided upon as depicted in the Table 1.

Table.1. Quantities and units

Variable	**Units**
Voltage	Volts
Current	μA
Resistance	MΩ
Capacitance	pF
Power	μW
Frequency	MHz

f. Cost function and constraint formulation

Sizing of the circuit requires the designer to decide the magnitude of C_c, Ws and Ls for all the transistors, branch currents I_5 and I_6, and bias voltage V_{gg}. It can be easily inferred that M1 and M2 have to be exactly matched as must be M3 and M4. Thus M1 and M3 can be replaced by M2 and M4 respectively. This reduces the dimensionality of the problem. Also the g_ms, I_Ds and W/Ls of the transistors are related. So for a given

transistor either two of these three can be selected as a variable. For the calculation of the auxiliary bias voltage V_{gg},

$$I_D = \frac{g_m}{2}(V_{GS} - V_T)\sqrt{1 + l\,V_{DS}} \cong \frac{g_m}{2}(V_{GS} - V_T) \quad \ldots \textbf{(47)}$$

can be used.

Thus the design variable space is much smaller than it appears to be.

A DC offset in the design should be avoided or at least reduced as far as possible. The DC offset the output voltage present without the input. So in order to make the DC operating point such that there is no DC offset, the simplified topology is considered, as shown in Fig.10.

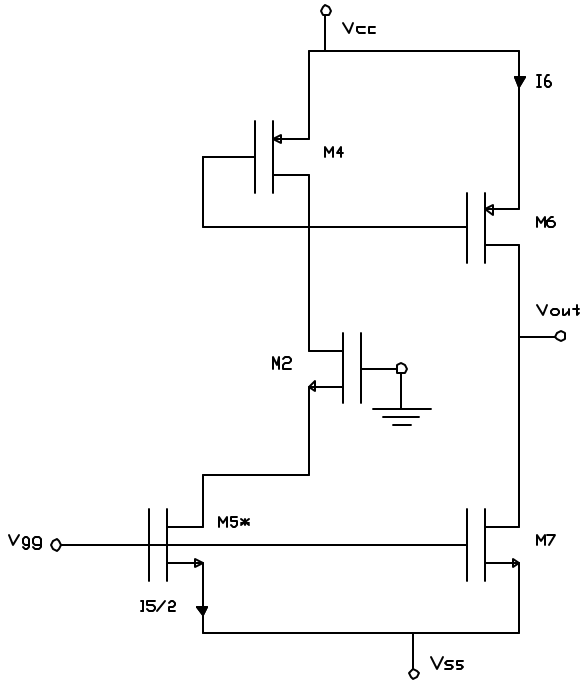


Fig.10. Topology for determining DC operating point

The gate-source voltages of the transistors M5* and M7 are the same, and likewise for M4 and M6. If the output voltage has to be zero when the input is absent, then the following relations must hold.

$$\frac{g_{m4}}{I_5/2} = \frac{g_{m6}}{I_6} \quad \ldots \textbf{(48)}$$

$$\frac{g_{m5}}{I_5} = \frac{g_{m7}}{I_6} \quad \ldots \textbf{(49)}$$

The independent variables chosen are g_{m2}, g_{m5}, g_{m6}, I_5, I_6 and C_c. Thus the design is restricted to be a six variable problem even if at the first glance the number of variables seems to be much higher.

The constraints are directly designed from the expressions used to define them and using the variables fixed for the problem formulation. The cost function can be area which can be adequately represented as the sum of the W/L ratios of the transistors, or it can be power dissipation or DC gain. The final design values returned primarily depend on the initial guess and the cost function, since the constraints are the same for all the different design objectives.

For example, the minimum output voltage is given by $V_{ss} + 2I_6/g_{m7}$. But the design variables does not include g_{m7}, hence it is necessary to express g_{m7} in terms of the design variables viz., $g_{m7} = (g_{m5} I_6)/I_5$.

Hence

$$V_{out(\min)} = V_{SS} + \frac{2I_6}{\left(g_{m5}\frac{I_6}{I_5}\right)} \quad \ldots \textbf{(50)}$$

Similarly all other constraints are converted from their regular forms to the required variable set.

6. Folded Cascode Operational Amplifier

a. Description of the circuit

All amplifier circuits can be reduced to the simple relation between the gain, transconductance and output impedance.

Gain = $g_m R_o$ … **(1)**

The gain can thus be increased by making the output impedance larger[25]. This is exactly the idea used in a cascode opamps. But in a simple telescopic cascode stage there is an inherent problem that the increase in gain is at the cost of reduction in the input range since the cascode stage comes directly inline with the gain amplifiers. A method used to overcome this hurdle is by using transistors of the opposite type as the cascode stages, thus they form a separate branch leaving the input range unchanged. There is always a price to pay for improving any circuit performance in analog design. The cascode stages obviously need more transistors and hence larger area.

In Fig.11, transistors M12 through M19 determine the common mode feedback output to be fed to the amplification stage. In this thesis only the sizing of the gain stage or the amplification stage has been considered. This stage is made up of M1 through M11assuming that the bias stage does not much affect the AC characteristics of the amplification stage.

From the topology it can be found out that the transistor pairs, M1 and M2, M4 and M5, M6 and M7, and M8 and M9, must have the same size.

b. Mathematical Equations

The performance specifications are given as

$$DC\ Gain = \frac{g_{m1}}{\left(\frac{g_{o8}\, g_{o10}}{g_{m8}} + \frac{\left(g_{o1} + g_{o4}\right) g_{o6}}{g_{m6}} \right)} \quad \ldots \textbf{(2)}$$

$$GB = \frac{g_{m1}}{2p\ C_L} \quad \ldots \textbf{(3)}$$

$$PM = \frac{180}{p} \tan^{-1}\left(\frac{g_{m6} C_L}{C_{L1} g_{m1}} \right) \ldots \textbf{(4)}$$

$$SR = \frac{I_{SS}}{2C_L} \quad \ldots \textbf{(5)}$$

$$Power = \left(2\, I_{casc} + I_{SS}\right)\left(V_{CC} - V_{SS}\right) \quad \ldots \textbf{(6)}$$

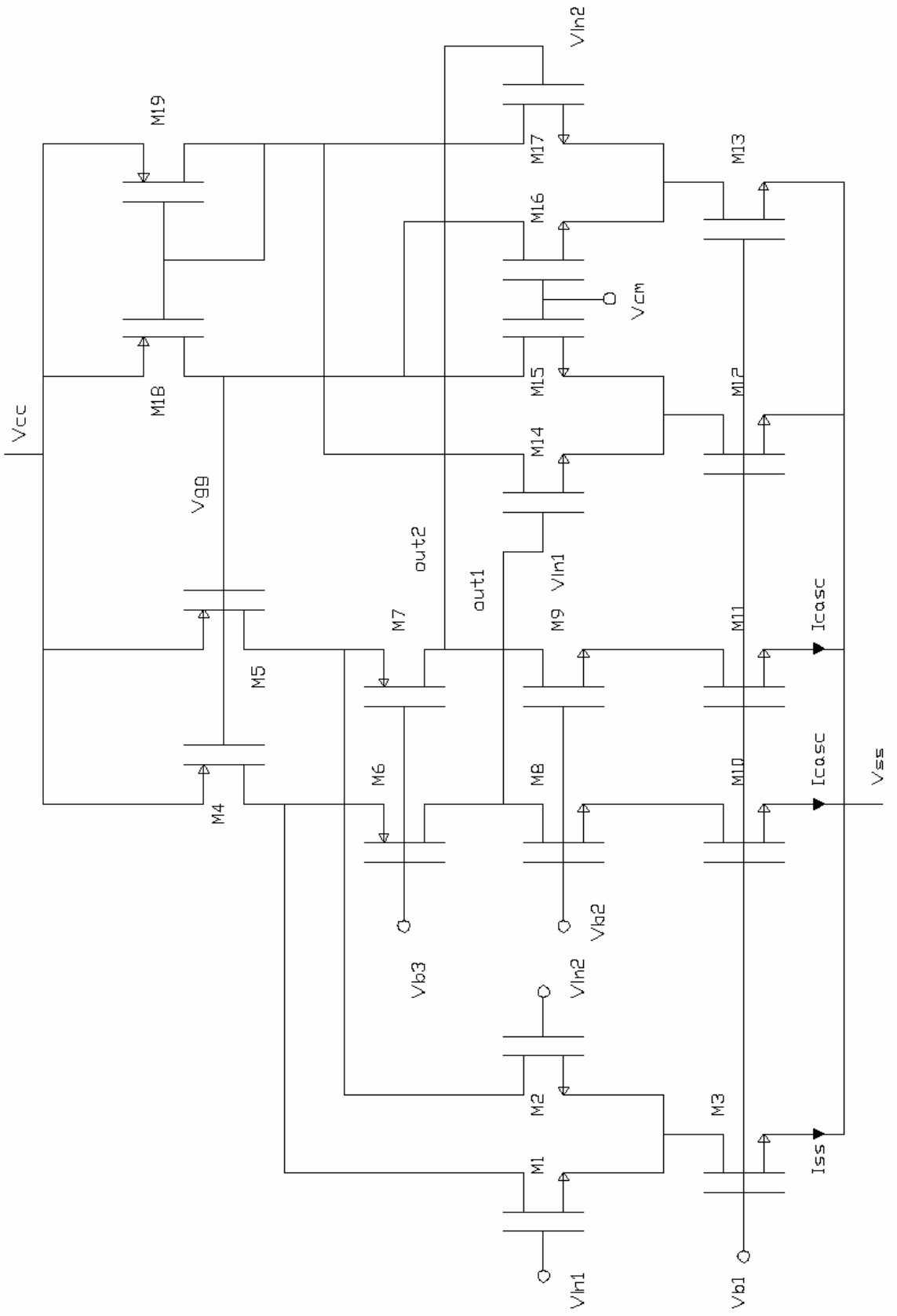


Fig.11. Fully Differential Folded-Cascode Op-amp

$$V_{out(\max)} = V_{CC} - \frac{2I_{casc}}{g_{m6}} - \frac{2I_{casc}}{g_{m4}} \quad \ldots \textbf{(7)}$$

$$V_{out(\max)} = V_{SS} + \frac{2I_{casc}}{g_{m10}} + \frac{2I_{casc}}{g_{m8}} \quad \ldots \textbf{(8)}$$

$$CMR_{\max} = V_{CC} - \frac{I_{SS}}{g_{m4}} + V_{TP} + V_{TN} \quad \ldots \textbf{(9)}$$

$$CMR_{\min} = V_{SS} + \frac{I_{SS}}{g_{m1}} + V_{TN} + \frac{2I_{SS}}{g_{m3}} \quad \ldots \textbf{(10)}$$

In the above relations $C_{L1} = C_{gd4} + C_{db4} + C_{db1} + C_{gs6} + C_{gd6} + C_{gd1}$ … **(11)**

The optimization strategy is the same as that for the two stage op-amp. In this case the variables are chosen to be I_{ss}, I_{casc}, g_{m1}, g_{m3}, g_{m4}, g_{m6}, and g_{m8}. Constraints are formed using the same methods.

c. Optimization Approach

There is a very subtle difference between the approach to two-stage and folded cascode opamp optimization. In case of a two-stage opamp, it behaves almost like a convex problem. Hence to generate more solutions, the cost function is varied. In case of a folded cascode opamp, the problem is highly non-linear and hence is very sensitive to the choice of initial points. This behavior is taken into consideration and the initial points are varied to give the user a greater selection of solutions.

d. Cost function and constraints formulation

The cost function and constraints are formed in exactly the same manner as the two-stage opamp.

7. Results and Discussion

The optimization problems were tackled with two different approaches. The two stage opamp was designed by varying the cost function and the sizes of the transistors for the folded cascode opamp were decided by changing the initial guess. This is done because the two stage opamp behaves like a convex problem with the given set of equations and constraints. Hence the starting point for the optimization process does not make any difference and solution converges to the same point. Whereas, for the folded cascode case, the problem is highly non-linear and non-convex, making it imperative to find out at least a few different design possibilities for the designer.

a. Two Stage OpAmp

For the purpose of testing the software, the following design specifications were considered.

DC Gain = 4000; Gain Bandwidth = 1MHz; Power Dissipation = 1000 μW;

Slew Rate = 2 V/μs; Minimum Output = -2V; Maximum Output = 2V;

Minimum CMR = -1V; Maximum CMR = 1V; Phase Margin = 45º

Other related parameters were fixed as follows.

Positive supply voltage V_{dd} = 2.5V

Negative supply voltage V_{ss}=-2.5V

Load Capacitance CL = 5pF

Load Resistance RL = 1MΩ

Results for variation in Size and DC gain multipliers

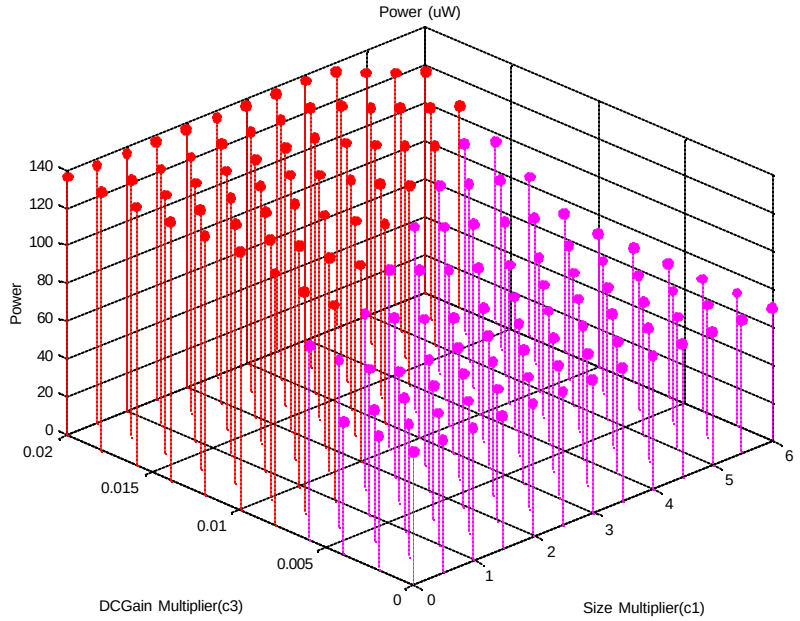


Fig.12. Power vs. DC Gain and Size Multipliers

Fig .12 shows that the minimum power possible for these design specifications is 70 μW. Also as the magnitude of the DC Gain multiplier increases for a constant size multiplier, the power dissipated keeps on increasing. This is due to the fact that both of them are proportional to I_6, in a way. The increase in power is much faster when the size multiplier is small in magnitude. As the size multiplier grows in magnitude, this rate of change in power is gradual.

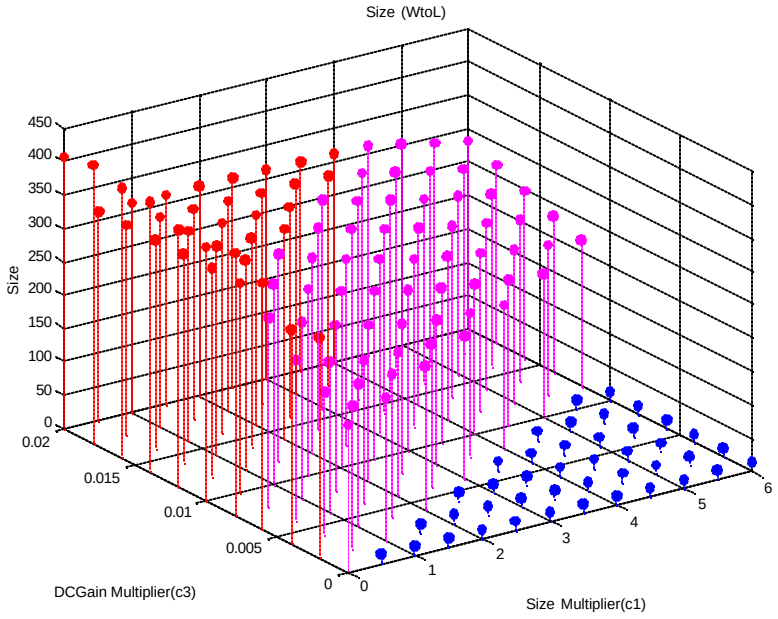


Fig.13. Size vs. DC Gain and Size Multipliers

Fig.13 shows more varied distribution than the corresponding power plot. The size in this algorithm is given by the summation of the W to L ratios of all the transistors. The minimum sum is seen to be around 13 and the maximum is seen around 400. The maximum size is reached when the size multiplier is zero and the DC gain multiplier is the highest. This is expected because, when the cost multiplier for the size is zero, the weight assigned to that in the overall cost function is zero. Hence it is the least effective of them all. For quite a few cost multiplier values, the smallest possible size of the circuit is obtained. The graph shows a very interesting characteristic regarding the change in values of size. The optimal size suddenly increases from around 25 to 250 for a small change in the DC Gain multiplier. The reason for this is switching constraints. Specifically the case of c_1=2.5 and c_2=0.004 and 0.006 was considered. For these values, a change in size from 13.8 to 214 was observed. The active constraints for the first set of

values are DC gain, slew rate, phase margin, minimum CMR and the lower bound on Cc was also reached. Whereas for the second set, the active constraints were slew rate, phase margin, minimum output, W to L ratio for M6 and the lower limit of Cc was again reached. The Lagrangian multipliers also changed their values for the common active constraints for the two cases. For instance, the multiplier for slew rate for case (1) was 61 which changed to 253 and that of phase margin was 0.75, which increased to 4.3. The increase in a constraint multiplier value indicates the bearing that the constraint has on the solution. So when the constraint having the highest value of multiplier is varied, it has the biggest impact on the solutions. These switching conditions cause the sudden changes in the optimum value.

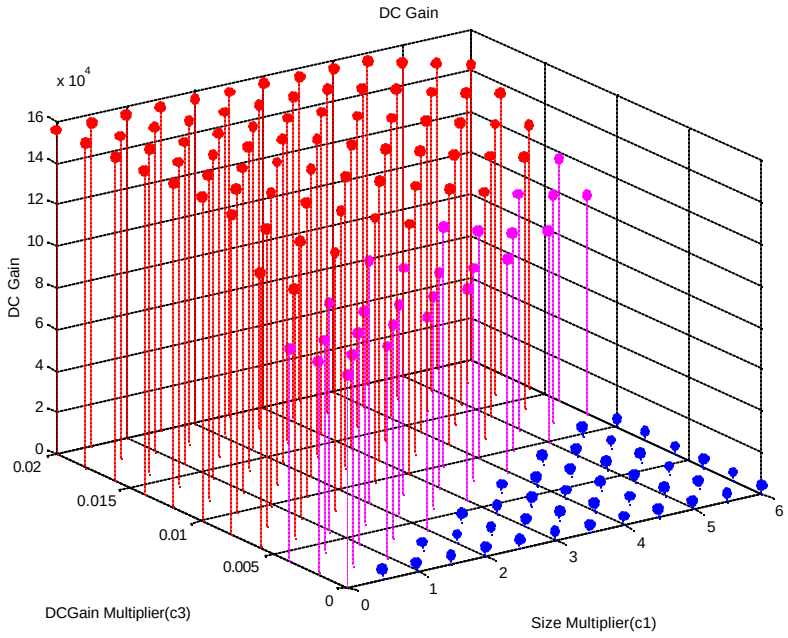


Fig.14. DC Gain vs. DC Gain and Size Multipliers

DC gain exhibits a plethora of different values as the multipliers vary. As can be logically deduced, the gain is highest when the multiplier related to it is high and other multipliers are smaller. The minimum possible gain is predominant in the region where gain multiplier is low and the size multiplier is quite high. The minimum possible gain given in the specifications is 4000. A condition similar to that of the size plot is seen in this plot in Fig.14 too. The change in DC gain from minimum to a very large number can also be attributed to the switching constraints.

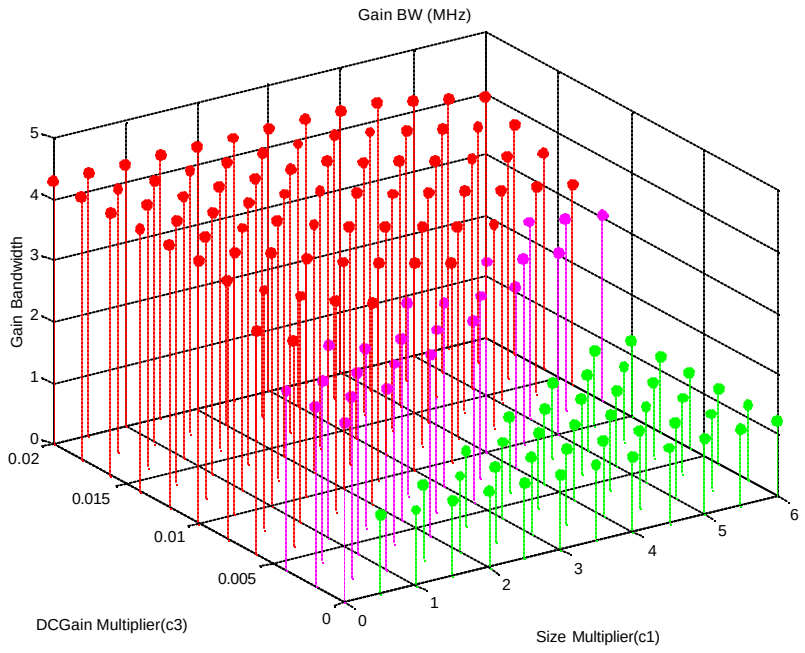


Fig.15. Gain Bandwidth vs. DC Gain and Size Multipliers

The trend seen in the gain bandwidth plot in Fig.15 is not different from the other three. For lower values of gain multiplier and higher size multiplier, the gain bandwidth latches on to the minimum limit. As the factor determining the relative weight of gain multiplier

in the combined cost-function increases, the gain bandwidth product increases. There is a twofold increase in the gain bandwidth product with the change in cost factors.

Analyzing all the plots at the same time gives a very good perspective of the entire opamp design process in specific and of any analog design process in general. For obtaining higher gain and higher bandwidth for a given set of specifications, a price has to be paid in terms of real estate and heat dissipation. This is because there is a tremendous increase in both size of the circuit and the power dissipated by the circuit. The slew rate, which is proportional to current, also increases with an increase in power, which is proportional to the square of current. It is almost impossible to overemphasize that there is no single best general solution for analog design. The keyword to dwell upon over here is “general”. Most of the analog design problems are highly specific and application oriented.

Results for variation in size and power multipliers

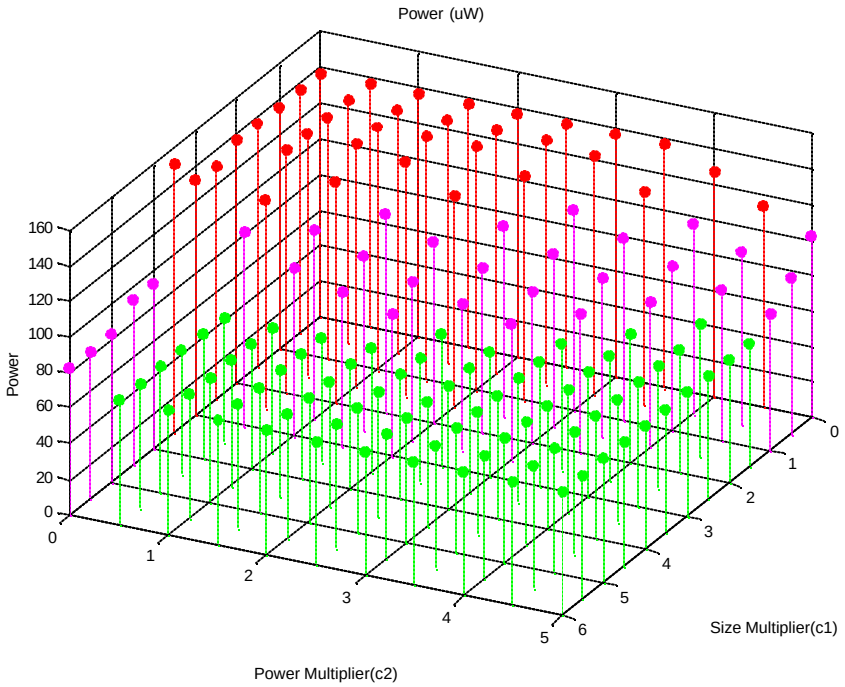


Fig.16. Power vs. Power and Size Multipliers

As can be seen in Fig.16, the power decreases with an increase in the power and size multipliers. At very low values of power multiplier, the power is quite high around 150 μW, but it reduces to about 70 μW as the cost multiplier increases in relative weight. For a constant power multiplier, the power dissipation increases with a reduction in the magnitude of the size multiplier. For higher power cost function factors, the power remains low for a longer time and then switches to its higher value. This can be deduced, because a higher factor for the power cost function translates into more relative weight for power in the overall cost.

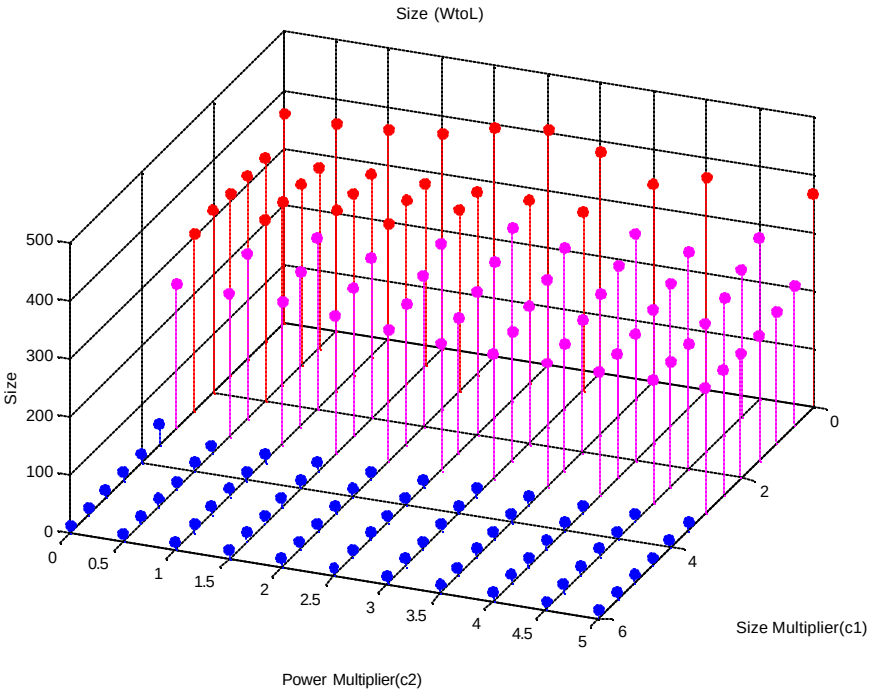


Fig.17. Size vs. Power and Size Multipliers

In Fig.17, the plot of size vs. multipliers reveals that the size remains quite small for a large range of multiplier values. This gives the designer an additional advantage of sorts because for a larger solution set, the size, which in many cases is the most important aspect of design, is quite small. On the whole, the size decreases with an increase in weight of size multiplier and power multiplier.

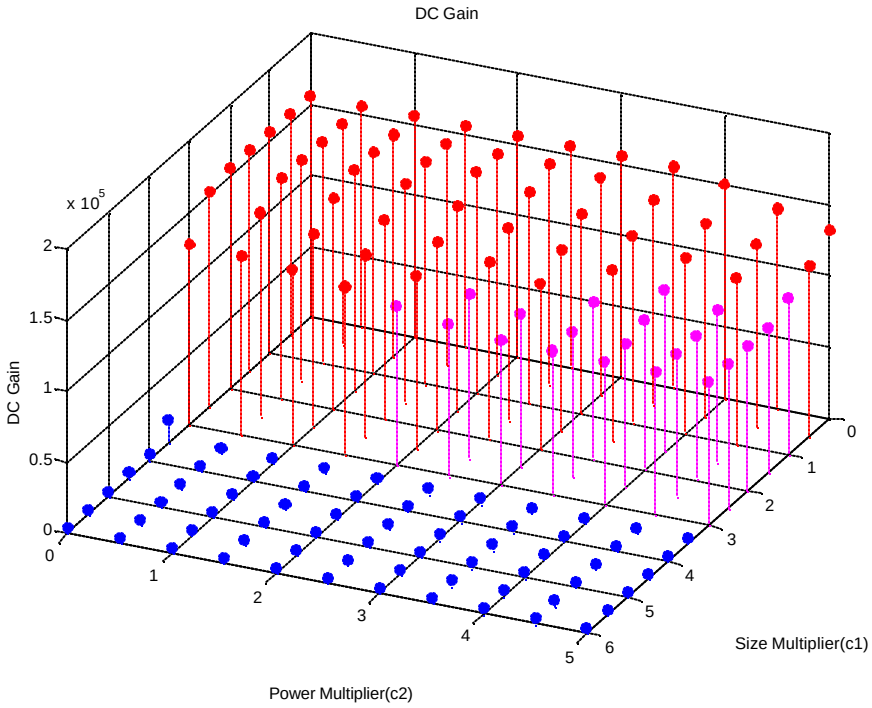


Fig.18. DC Gain vs. Power and Size Multipliers

The DC gain trend, as seen in Fig.18, is very similar to that of the plot of size. For all the points that show minimum size, corresponding points on the DC gain plot also demonstrate minimum allowable gain. This could also be thought of as a trade-off in the cost functions. For a higher DC gain, higher power dissipation has to be allowed in a larger circuit. When the specific values of c_1=3 and c_1=3.5, for c_2=2 were analyzed, as expected, switching constraints were observed. The minimum output and upper limit of W to L ratio for M6 switched off when c_1 was changed from 3 to 3.5. Conversely the constraints for CMR min and DC gain became active. Such switching constraints are responsible for the change in cost function values.

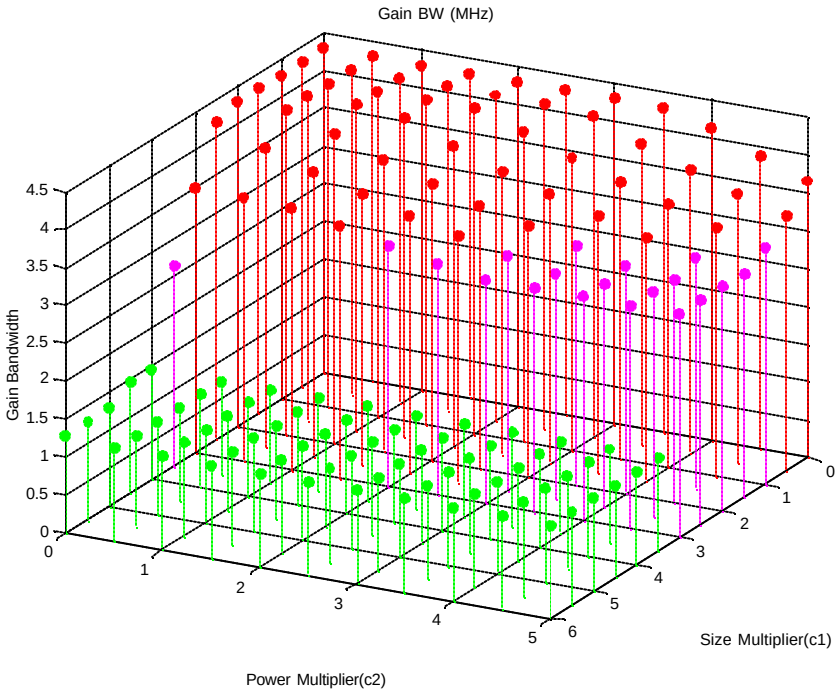


Fig.19. Gain Bandwidth vs. Power and Size Multipliers

The gain bandwidth plot, as shown in Fig.19, also exhibits one-to-one correspondence with the DC gain and size plots. The maximum possible value is seen to be about 4.2MHz and the lower limit is reached at the other end of the spectrum. The fluctuations in gain bandwidth have repercussions on the other cost functions. It grows in tandem with the dc gain and at the expense of power and size. As the weight assigned to the size decreases, its relative impact in the overall cost function reduces. Hence the size of the circuit increases, due to the increase in the transconductance of M2. This translates into a higher gain bandwidth product.

Results for variation in power and DC gain multipliers

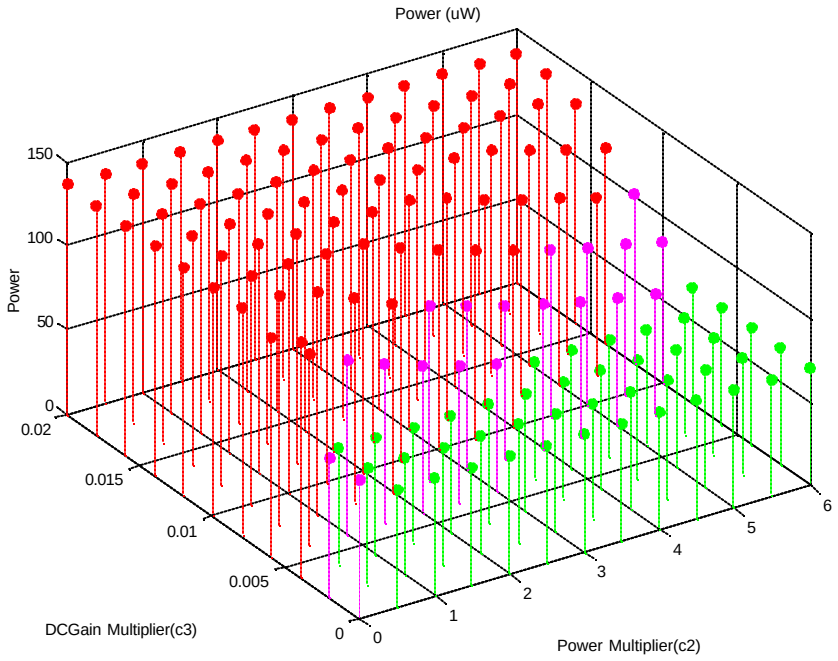


Fig.20. Power vs. DC Gain and Power Multipliers

As seen in Fig.20, the distribution of the power function over the power and DC gain multiplier axes has its minimum when the multiplier for power is high and that of the DC gain is low. The value increases as the power multiplier reduces or the DC gain multiplier increases. When two points were considered with c_2=1.5 and c_3=0.004 and c_2=1.5 and c_3=0.006, they too showed a tremendous change in the cost function components with such a small variation in the cost function multiplying factors. The gain increased from 72dB to 100dB, size increased from 13.8 to 250, and power dissipated from 70μW to 95μW. This stupendous change can be attributed to constraints which were inactive becoming active and some which were active becoming inactive. Apart from this, the Lagrangian multipliers of the constraints continuing to remain active did change too. For

instance, DC gain, which was latched on to the lower limit, increased by more than an order to make the DC gain constraint inactive, the slew rate constraint multiplier changed almost by a factor of 10 and so did that of the phase margin by more than a factor of 3, from .75 to 2.5. Also min CMR constraint became inactive, whereas that of minimum output and the upper limit of W to L ratio of M6 became active.

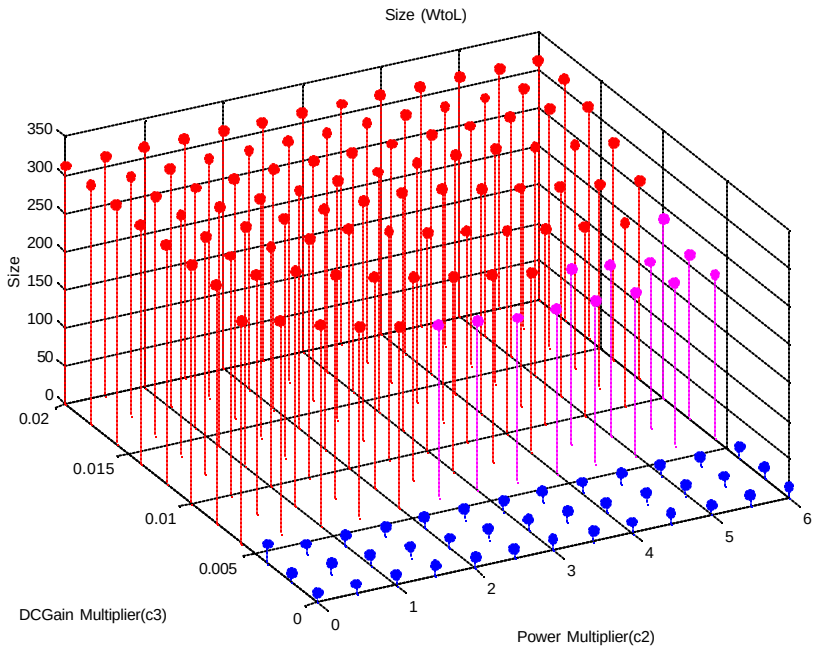


Fig.21. Size vs. DC Gain and Power Multipliers

Again in Fig.21, it can be observed that the summation of W to L ratios is quite low when the power multiplier is high. As the DC gain multiplier increases in value, the total W to L ratio tends to increase. The higher value is seen to be around 300. Power and size increase and decrease together.

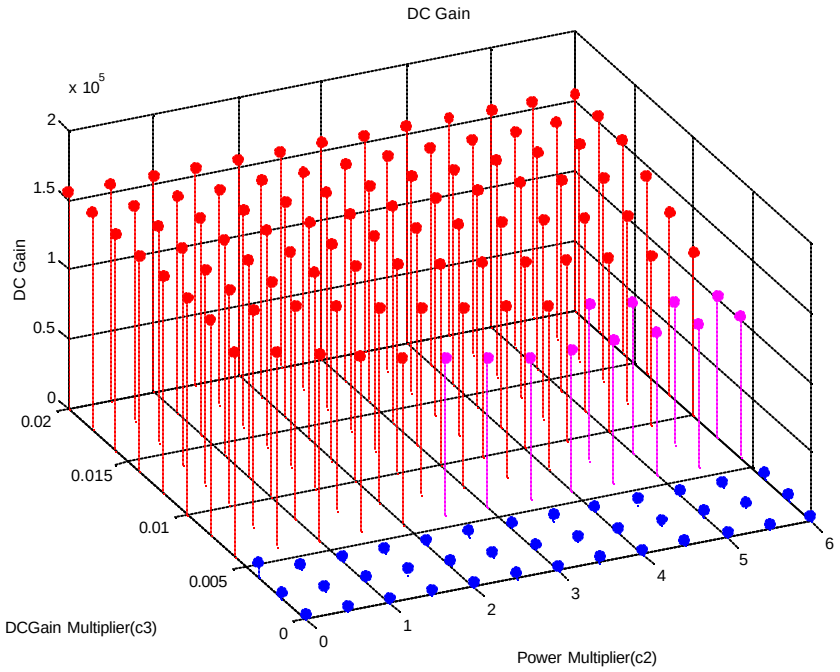


Fig.22. DC Gain vs. DC Gain and Power Multipliers

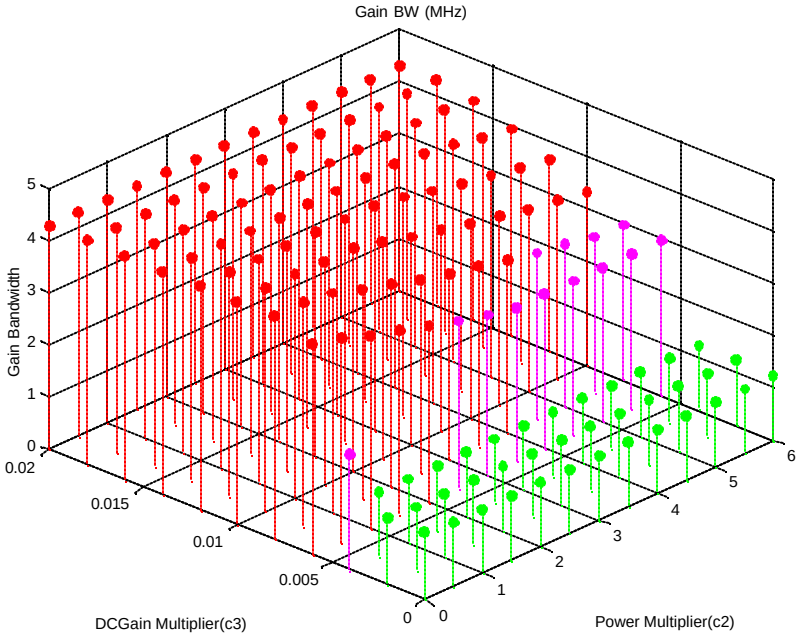


Fig.23. Gain Bandwidth vs. DC Gain and Power Multipliers

Similar trends are seen in the plot of DC gain and gain bandwidth, illustrated in Figs. 22 and 23. But an important point to consider here is that a designer would like to have higher DC gain and gain bandwidth, but rather keep the size and power low. Even if the trends are similar, they are opposite in nature, so far as the objectives are concerned. Hence analog design is usually a compromise between two conflicting cost functions.

SPICE Simulation

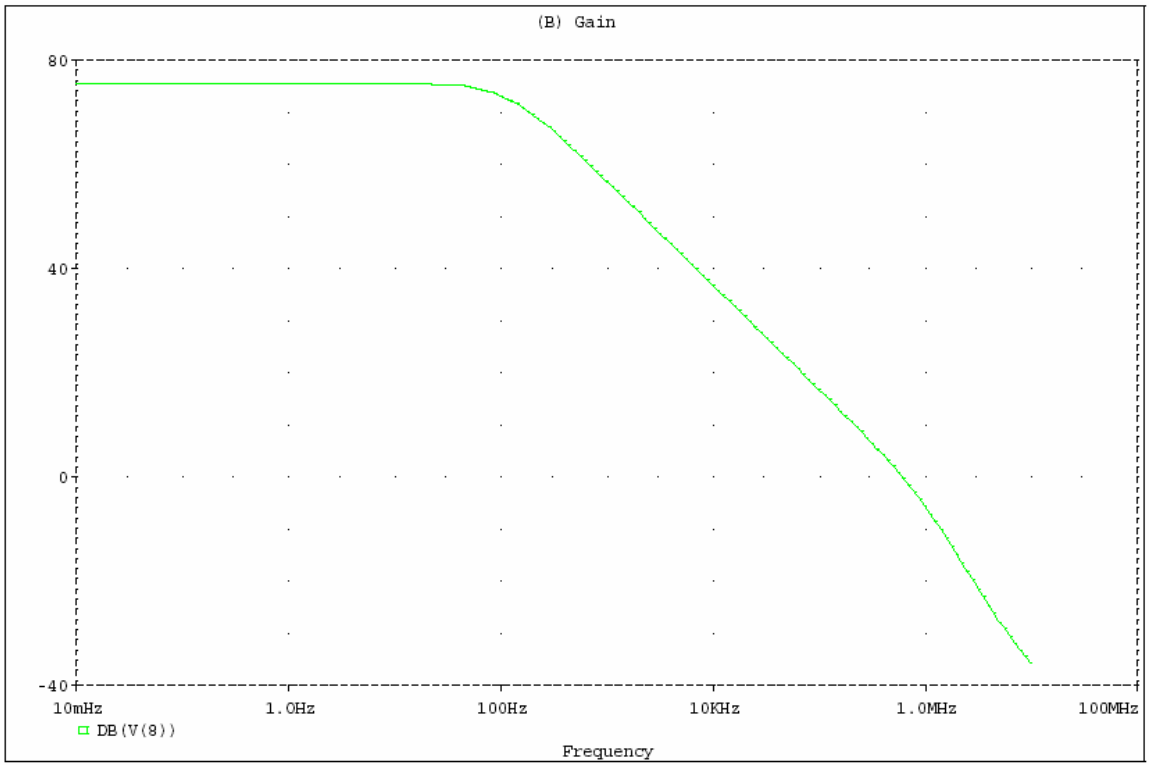


Fig.24. Gain plot of a two stage opamp

A certain point is chosen from the array of solutions and using those values, the circuit is verified using PSpice[28][30]. As seen in Fig.24, the DC gain is more than 70, which is the specified minimum. The sum of W to L ratios is 13.5.

b. Folded Cascode Opamp

Size, DC gain and Power

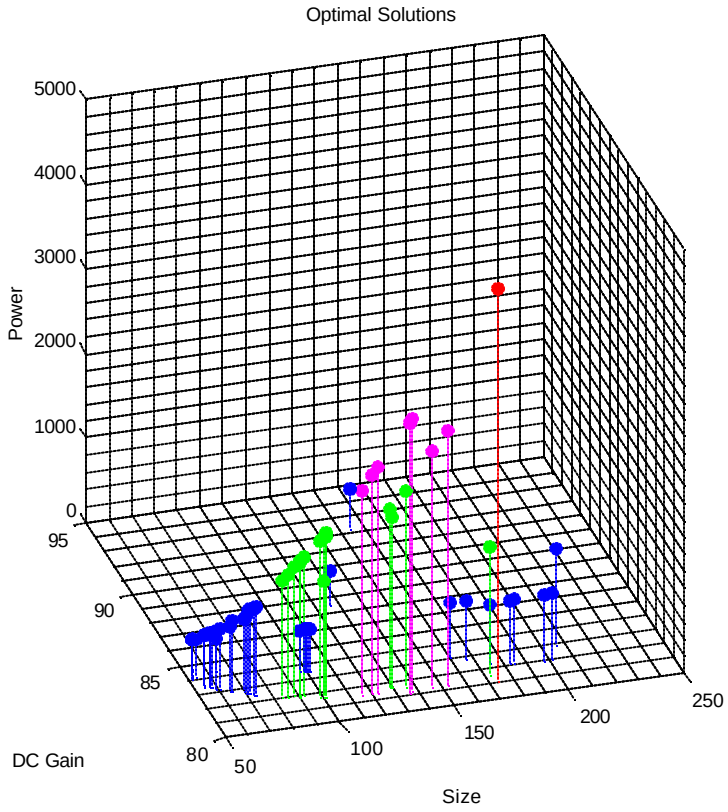


Fig.25. DC Gain vs. Size vs. Power

The plot in Fig.25 of DC gain vs. Size vs. Power does not really describe any trend because it is essentially a plot of three different parameters describing the folded cascode opamp circuit. That does not in any way mean that it is not insightful to analyze this graph. Another view of the same, shown in Fig.26, which can aid the process, is the one looking over the top of the box, so that Size forms the x-axis, DC gain forms the y-axis and Power on the z-axis is just seen as points on this graph. The color coding is such that, for the entire range of power values, the lowest one fourth of the range is blue, next is

green, next is magenta and the top one fourth is red. This distribution is not according to the number of points; it is according to the absolute value of power dissipation, because the decision of the designer is going to be based on it.

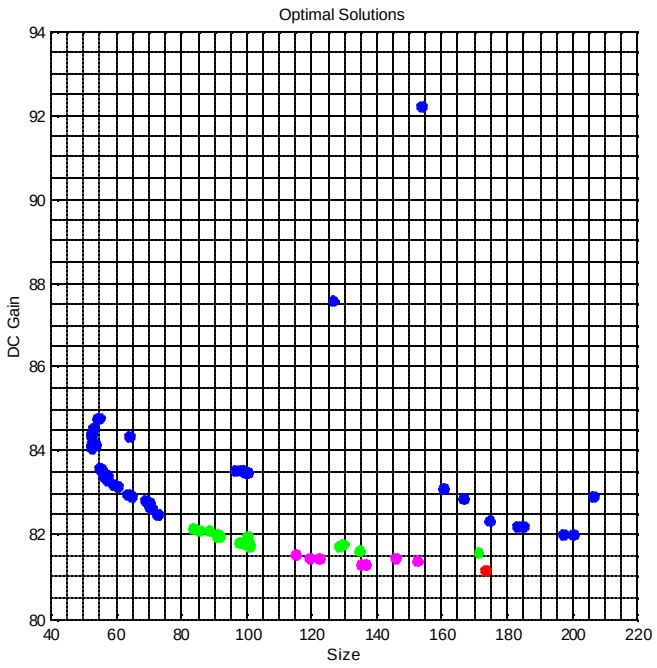


Fig.26. DC Gain vs. Size

These solutions have been obtained by changing the initial guess, the cost function, specifications remaining the same. The most primitive deduction from this plot is that the problem is non-linear. There are 68 different solutions obtained from 2187 initial guesses. A criterion for accepting solutions has been put in place so that the user is not overwhelmed by sheer numbers. Since the enhancement of any cost function is at the expense of some other, the maximum additional allowable cost is limited to thrice the minimum of the costs obtained. So in effect, if the cost function is Size, and the least size

value obtained is ‘x’ then an acceptable solution has a size of at the most ‘4x’. This keeps all the values within reasonable limits.

Other two dimensional views can clarify the distribution further.

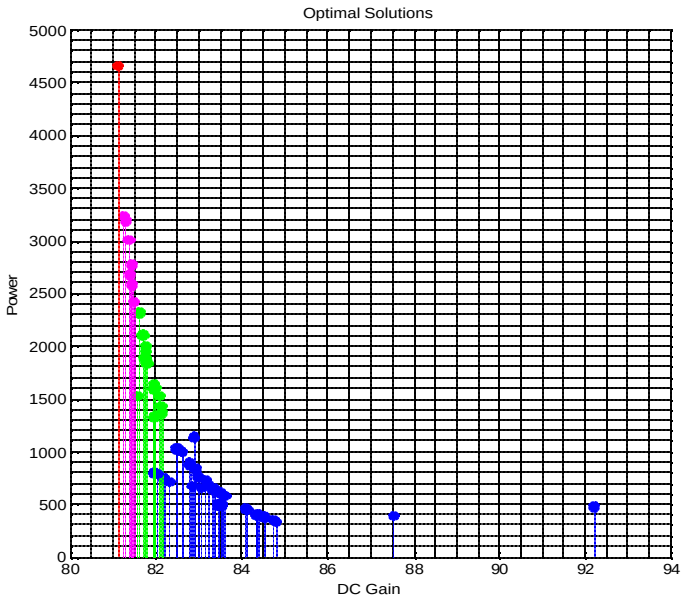


Fig.27. Power vs. DC Gain

As seen in Fig.27, the initial points make a lot of difference in the convergence of solutions. For some of the initial points, the DC gain is very low and power is quite high; which in turn means that the slew rate is going to be high for such a point. For some other solutions, the DC gain is high and the power dissipated is quite low. The formula for DC gain has the current terms in its denominator, whereas the power value is directly proportional to the current. Hence the distribution of the solutions is such that the terms having higher DC gain, have lower power.

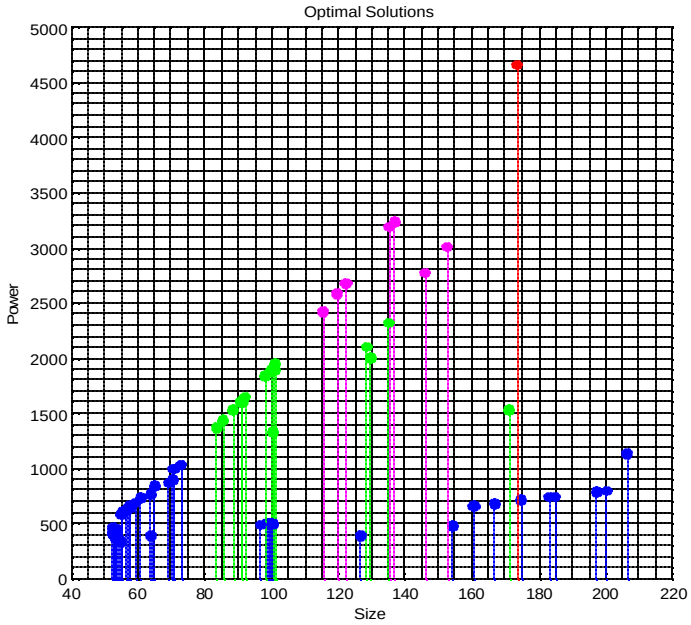


Fig.28. Power vs. Size

In the other plot, Fig.28, the distribution is quite informative. Many solutions have large sizes and smaller power. This is because the power is directly proportional to the currents and size is inversely proportional to them. Thus an increase in one is accompanied by a decrease in the other. But there are other factors responsible for the size value such as the transconductance values of the transistors. Hence there are quite a few points which have both lower power and lower size, but these points effectively have lower values for DC gain too.

Gain Bandwidth, Phase Margin and Slew Rate

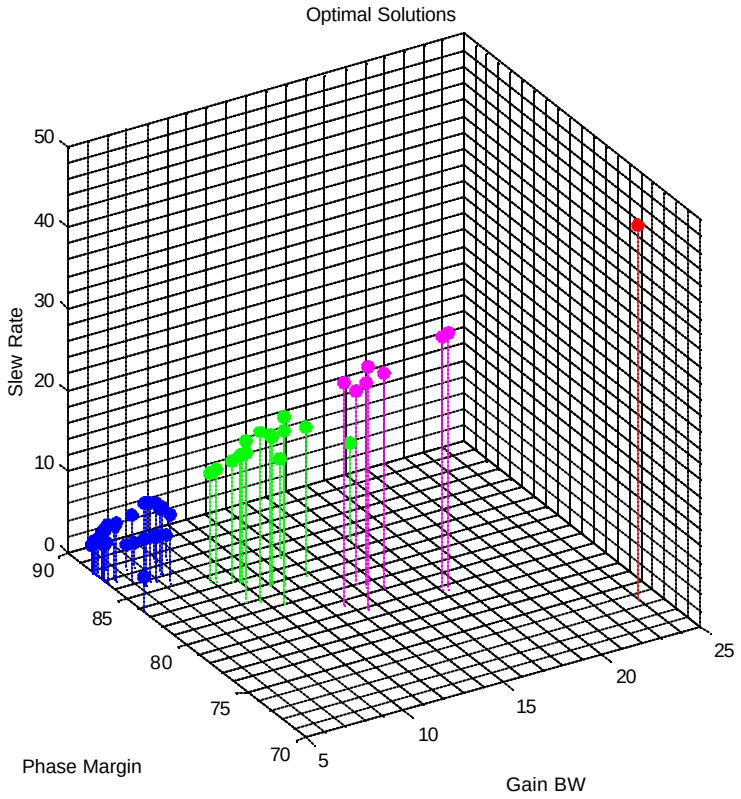


Fig.29. Phase Margin vs. Gain Bandwidth vs. Slew Rate

For the same set of 68 solutions, the next three functions plotted are gain bandwidth, phase margin and slew rate. This plot in Fig.29 exhibits more uniformity than the previous one. There are certain trends that can be more readily observed with the help of two dimensional projections of these three dimensional plots. The color coding is similar to that of the previous one, and the only difference being, the functions that are plotted. So the colors of the stem graph are proportional to the slew rate of the solution point.

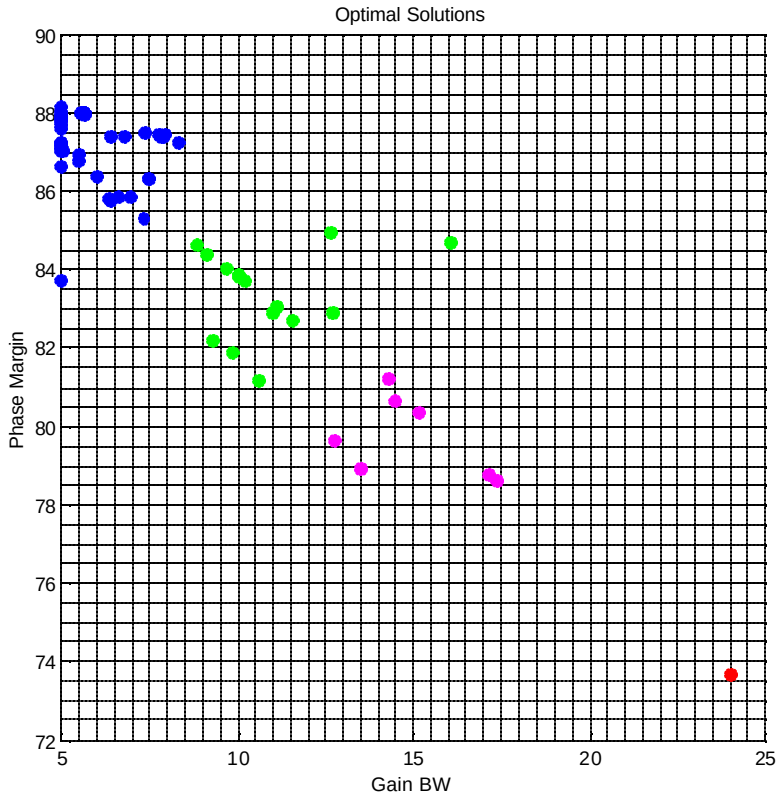


Fig.30. Gain Bandwidth vs. Phase Margin

In the "Top View" of the plot, illustrated in Fig.30, the distribution of points along the gain bandwidth and phase margin can be seen. If a trend line is drawn for the given points, it exhibits a negative slope, signifying that the phase margin and gain bandwidth are functions that increase in the opposite directions. This can be attributed to their dependence on the transconductance value of M1. Increase in g_{m1} causes an increase in gain bandwidth, whereas a reduction in phase margin.

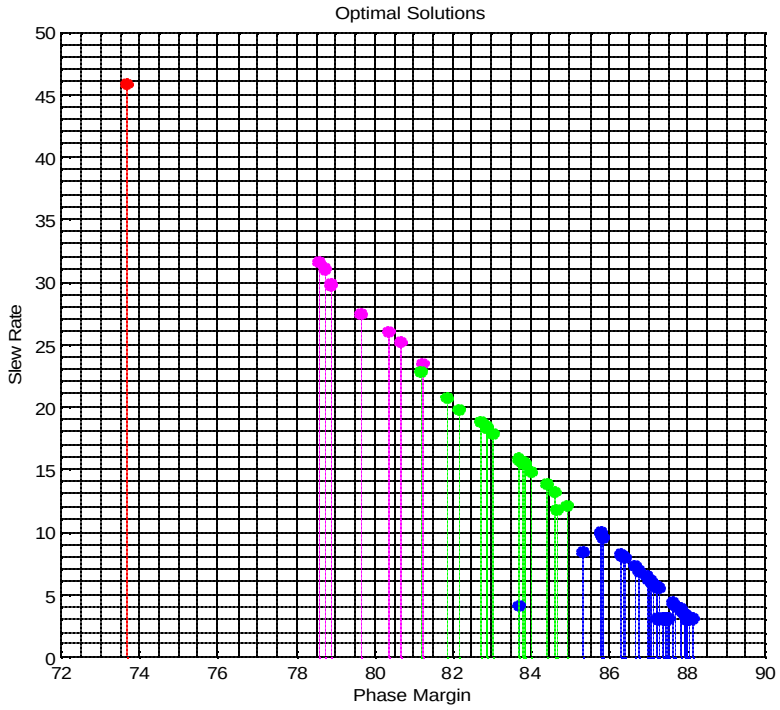


Fig.31. Phase Margin vs. Slew Rate

In Fig.31, slew rate and phase margin too can be seen to be increasing in the opposite directions. Hence a solution having smaller phase margin has a higher slew rate; which can be considered to be a representative of the power dissipation of the opamp. The phase margin is inversely proportional to the transconductance of M2 and the slew rate is proportional to current. As g_{m2} reduces to decrease the size, the current too drops. This causes a reduction in slew rate explaining the negative slope of the trend line.

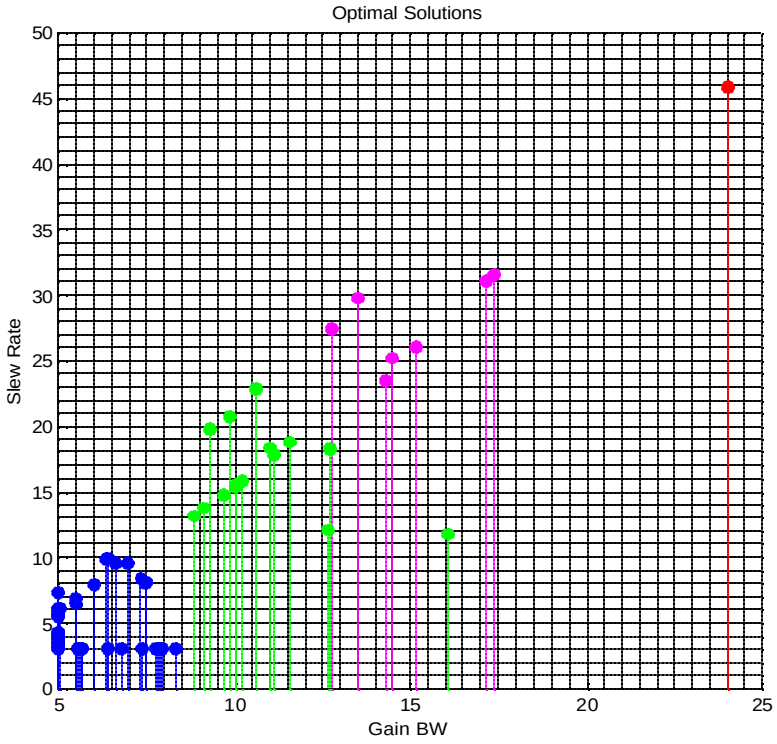


Fig.32. Gain Bandwidth vs. Slew Rate

The gain bandwidth and the slew rate are seen to be increasing together in Fig.32. They are complementary functions, because it is desirable to have higher values for both of them, and it is possible to have such a solution.

8. Conclusion and suggestions for future work

This work is just one of the ways in which the entire gamut of analog circuits has been tackled. The problem decided beforehand was successfully tackled with reasonably good results. This thesis was an attempt to provide a tool for optimizing opamp circuits to be used by analog designers. In that respect the goal was accomplished. This program is able to provide a GUI for the user so that the circuit specifications can be easily changed. It optimizes a circuit and provides many design alternatives to be chosen from. Then any particular design can be verified by using SPICE, because the package has the ability to generate a SPICE netlist. Thus it provides a complete "tool" to the designer of an opamp. But there is always a room for improvement. Some of the changes that can be suggested are

1) The transistor model used in this case is the level 1 model. To get more accurate results, a more developed model such as the BSIM or EKV model can be used.

2) Now that the basic skeletal design of the algorithm has been verified to be working for two topologies, more can be added to the same.

3) More intelligence can be added to the software after adding more topologies, so that the program is in a position to choose the correct one for the given problem.

9. References

[1] Aguirre, I. & Carlosena, A. Automated simplified analysis of analogue circuit behavior based on fully symbolic relations between parameters. *International Journal of electronics*, vol. 88(10), pp.1103-1115. (2001).

[2] Allen, P. & Holberg, D. *CMOS Analog Circuit Design*, New York: Holt, Rinehart & Winston. (1987).

[3] Arora, J. *Introduction to Optimum Design*, New York: McGraw-Hill. (1989).

[4] Baker, R.J. & Li, H.W. & Boyce, D. *CMOS circuit design, layout and simulation*, New York: IEEE Press. (1998).

[5] Becerra, J. & Friedman, E. Analog design issues in digital VLSI circuits and systems. *Analog Integrated Circuits and Signal Processing*, vol. 14, pp.5-8. (1997).

[6] Bucher, M. et al. An efficient parameter extraction methodology for the EKV MOST model. *Proceedings of the IEEE international conference on microelectronic test structures*, vol. 9, pp.145-150. (1996).

[7] Cabeza, R. & Carlosena, A. On the use of symbolic analyzers in circuit synthesis. *Analog Integrated Circuits and Signal Processing*, vol. 25, pp.67-75. (2000).

[8] Carley, L. et al. Synthesis tools for Mixed-Signal ICs: Progress on frontend and backend strategies. *33rd Design Automation Conference*. (1996).

[9] Cohn, J. et al. KOAN/ANAGRAM II: New tools for device-placement and routing. IEEE Journal of solid state circuits, vol. 26. (1991).

[10] Enz, C. et al. An analytical MOS transistor model valid in all regions of operation and dedicated to low-voltage and low-current operations. *Analog Integrated Circuits and Signal Processing*, vol. 8, pp.83-114. (1995).

[11] Fernández, F. Guerra, O. Rodriguez-Garcia, J. & Rodriguez-Vazquez, A. Symbolic analysis of large analog integrated circuits: The numerical reference generation problem. *IEEE Transactions on circuits and systems-II: Analog and Digital Signal Processing*, vol. 45(10), pp.1351-1361. (1998).

[12] Funaba, S. et al. A fast and accurate method of redesigning analog subcircuits for technology scaling. *Analog Integrated Circuits and Signal Processing*, vol. 25, pp.299-307. (2000).

[13] Geppert, L. Electronic Design Automation. *IEEE Spectrum*, pp.70-74. (January 2000).

[14] Gielen, G. Symbolic analysis methods and applications – An overview. *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 1141-1144. (1992).

[15] Gielen, G. & Sansen, W. *Symbolic analysis for automated design of analog integrated circuits*, Boston: Kluwer Academic Publishers. (1991).

[16] Gielen, G. et al. Analog circuit design optimization based on symbolic simulation and simulated annealing. *IEEE Journal of solid state circuits*, vol. SC-25(3), pp.707-713. (1990).

[17] Gielen, G. et al. ISAAC: A symbolic simulator for analog integrated circuits. *IEEE Journal of solid state circuits*, vol. SC-24(6), pp.1587-1597. (1989).

[18] Gray, P. & Meyer, P. *Analysis and design of analog integrated circuits*, New York: John Wiley & sons. (1993).

[19] Gray, P. & Meyer, R. MOS Operational Amplifier Design – A Tutorial Overview. *IEEE Journal of Solid-State Circuits*, vol. SC-17(6), pp.969-982. (1982).

[20] Hershenson, M. Boyd, S. & Lee, T. CMOS Operational Amplifier Design and Optimization via Geometric Programming. *Proceedings of the first international workshop on Design of Mixed-Mode integrated circuits and applications*, pp.15-18. (1997).

[21] Johns, D. & Martin, K. *Analog Integrated Circuit Design*, New York: John Wiley & sons. (1997).

[22] Jusuf, G. et al. CADICS: Cyclic analog-to-digital converter synthesis. Proceedings of IEEE Conference on Computer-Aided Design, pp.286-289. (1990).

[23] Koh, H. et al. OPASYN: A compiler for CMOS operational amplifiers. *IEEE Transactions on CAD*, vol. 9(2), pp.113-125. (1990).

[24] Kruiskamp, W. & Leenaerts, D. DARWIN: CMOS opamp Synthesis by means of a genetic algorithm. *32nd ACM/IEEE Design Automation Conference*. (1995).

[25] Mallya, S. & Nevin, J. Design Procedures of a Fully Differential Folded-Cascode Operational Amplifier. *IEEE Journal of Solid-State Circuits*, vol. 24(6), pp.1737-1740. (1989).

[26] Mandal, P. & Visvanathan, V. CMOS Op-Amp sizing using a geometric programming formulation. *IEEE Transactions on Computer-Aided Design of Integrated circuits and systems*, vol. 20(1), pp.22-38. (2001).

[27] Marchand, P. *Graphics and GUIs with MATLAB*, Boca Raton FL: CRC Press. (1996).

[28] Monssen, F. *MicroSim PSpice with circuit analysis*, Upper Saddle River, NJ: Prentice Hall. (1998).

[29] Ochotta, E. et al. Synthesis of high performance analog circuits in ASTRX/OBLX. *IEEE Transactions on Computer-Aided Design*, vol. 15, pp.273-293. (1996).

[30] Rashid , M.H. *SPICE for circuits and Electronics using PSpice*, Englewood Cliffs, NJ: Prentice Hall. (1990).

[31] Shi, C.-J.R. & Tan, X-D. Canonical symbolic analysis of large analog circuits with determinant decision diagrams. *IEEE Transactions on Computer-Aided Design of Integrated circuits and systems*, vol. 19(1), pp.1-18. (2000).

[32] Shi, C-J.R. & Tian, M. Simulation and sensitivity of linear analog circuits under parameter variations by robust interval analysis. *ACM Transactions on Design Automation of Electronic Systems*, vol. 4(3), pp.280-312. (1999).

[33] Tsividis, Y. & Suyama, K. MOSFET Modeling for Analog Circuit CAD: Problems and Prospects. *IEEE Journal of Solid-State Circuits*, vol. 29(3), pp.210-216. (1994).

[34] Ye, J. *Automated transistor sizing for analog circuit based on symbolic simulation*, M.S. Thesis: University of Cincinnati. (1997).

[35] Zebulum, R. et al. A multi-objective optimization methodology applied to the synthesis of low-power operational amplifiers. *Proceedings of the 13th conference in Microelectronic and circuit packaging*, vol. 1, pp.264-271. (1998).

Websites:

http://www.ece.utexas.edu/~holberg/lecture_notes/bk3.pdf

Visual Basic

http://www.vbinformation.com/tutor.htm

http://matlab_world.myetang.com/matlab_activex_e.htm

http://www.vbsquare.com/files/

http://216.26.168.92/vbworld/default.aspx

http://www.martin2k.co.uk/phorum/read.php?f=1&i=254&t=254

http://www.other-space.com/vb/part2/multipleforms.html

Matlab

http://www.mathworks.com

http://www.rit.edu/~pnveme/Matlab_Course/Matlab_HandleGraphics.html

Symbolic analysis

http://www.eng.uts.edu.au/~benr/

http://www.dice.ucl.ac.be/~flandre/SMACD96/smacd96.html

SPICE

http://www.ee.byu.edu/support/Spice/pspice.htm

http://www.eecs.umich.edu/~stevenmm/academics/courses/413/lab3.pdf

APPENDIX

Instructions for the user

1) Start the software by either double-clicking on the executable ".exe" file or opening the project in Visual Basic environment and then clicking on the run button.
2) This should open a window which will have fields to accept the specifications. This window also has five buttons labeled as "Submit", "Reset", "Optimize", "Write Spice File", and "Launch PSpice".
3) After entering the specification values (hit the "Reset" key if default values are needed), click on "Submit". This will open a text input box that asks for a path where all the files are stored. Remember to put a '\' at the end of the path. For example, if the path is a directory called "users" in the C drive, then enter the path as "c:\users\". A message displaying the path will be shown after which another message asking the user to hit "Optimize" will be shown.
4) Hit "Optimize" to run the optimization routine which should start Matlab.
    a. In case of a two stage opamp optimization, at this point the software will ask the user to choose among three sets of cost functions, viz. DC gain and size, DC gain and power, and size and power. Choose the pair of functions whose relative weight needs to be varied in the overall cost function. Clicking on one of these three should start the actual optimization routine. The message "Simulation in Progress" will be displayed on the screen.
    b. In case of the folded cascode opamp, as soon as "Optimize" is clicked, Matlab starts executing the optimization routine with different initial

points. The message “Simulation in Progress” will be displayed on the screen.

This can take a while depending on the number of steps in cost function multiplying factors in case of two stage opamp and the number of initial points in case of folded cascode opamp.

5) At the end of the optimization, Matlab will plot graphs and will write the data into Excel files. “Solutions.xls” for two stage opamp and “FCsolutions.xls” for folded cascode opamp. **Note: This is done every time “Optimize” is clicked, so whenever an Excel file is written again, it will display a message saying “solutions.xls exists. Do you want to overwrite?”; hit “Yes” to overwrite, because it is necessary that the data be stored in “solutions.xls”/ “FCsolutions.xls” for the proper functioning of the program.**
6) In the case of folded cascode opamp, after the Excel file is written, an option window pops up, where the functions to be plotted against each other can be chosen. The choice is between “Power vs. DC Gain vs. Size” and “Slew Rate vs. Gain Bandwidth vs. Phase Margin” in the case of folded cascode optimization.
7) There should be plots on the screen in either case at this point. Also a set of guidelines is displayed on screen. These plots are basically three-dimensional stem3 plots from Matlab which have been displayed in the two-dimensional format.
    a. In case of two stage opamp, these plots have the two cost function multipliers as the x and y axes and a function as the z axis. These have been color coded such that the entire range of the cost function being plotted is divided into four

parts. The smallest $1/4^{th}$ are colored blue, the next $1/4^{th}$ are colored green, next $1/4^{th}$ are magenta and the highest quarter of them are red. Note that this division is in terms of the value of the cost function and not that of the number of points in that range. The graph can be rotated in 3-d to analyze the behavior of the cost functions. To get the original 2-d view of the plot, bring the concerned plot window to the top. For instance if "size" is being analyzed, then bring that window to the top, and then select the Matlab command window, and type in view(0,90) in the command window or else rotate the plot till the desired view is obtained.

b. In case of the folded cascode opamp, "DC gain and size form the x and y axes and power forms the z-axis" and "phase margin and gain bandwidth form the x and y axes, and slew rate forms the z-axis". In this case too the color coding is the same, in the sense that the z-axis is divided into four parts and so forth. This plot is also a stem3 plot and can be manipulated in the same manner as that of the two stage opamp.

8) Any of the points generated on the plots can be taken into consideration for simulation using PSpice. To this end a point must be selected on the plots. It has been seen that it is easiest to select a point with the "view(0,90)". This is because when a point is selected, the x and y coordinates are stored by the program. And this correspondence between user visualization and computer interpretation can be the best with this view. To select any point, first click inside the window once, this should change the arrow to a set of perpendicular lines. Wait for a few seconds if the change does not occur immediately. Once the perpendicular lines

appear, place the cross on the point of interest. The lines should change to an arrow again. When this happens, the point is correctly realized and stored for the purpose of simulation.

9) After the point is chosen, go back to the main window with the five buttons. Click on the button labeled “Write Spice File” to create a SPICE netlist for the selected point. This will ask for a path to store the file and then give an acknowledgement after having written the file.
10) To launch the PSpice program, click on the button labeled “Launch PSpice”. Then the file created can be verified by simulation.

The following files are a part of the entire package.

1) Matlab Files

a) Common files

“Simulation in progress” files: siminprog.fig, siminprog.m

“Guidelines” file: twostagehelp.fig, twostagehelp.m

b) Two stage opamp

Main optimization files: meshlinear.m, meshlinear12.m, meshlinear23.m

Cost function file: combfun1.m

Constraint functions file: projnlcon1.m (This file may be absent. This is written by VB during the program execution)

File for the window providing option between cost functions: twostageoption.fig, twostageoption.m

c) Folded cascode opamp

Main optimization file: fc10per.m

Cost function file: foldcasfun.m

Constraint functions file: foldcascon.m

File for the window providing the option between different functions to plot:

foldcasoption.fig, foldcasoption.m

Files for plotting functions:

BW_SR_PM.m (gain bandwidth, slew rate and phase margin)

trialplot.m (power, DC gain and size)

2) Visual Basic Files

a) Two stage opamp

finalproj.vbp, constraint.frm and other associated files.

b) Folded cascode opamp

foldcasgrp1.vbp, mainform1.frm and other associated files.

Tweaking the program

a) Two Stage Opamp

A few important places where the program can be tuned are as follows:

The meshlinear, meshlinear12 and meshlinear23 files basically scan the solution space for changing cost multipliers. The multiplier for size is c_1, power is c_2, DC gain is c_3 and gain bandwidth is c_4. The range of this cost multipliers depends on the magnitude of the individual function. For example the DC gain is in thousands whereas the gain bandwidth

never goes beyond single digits. This disparity requires the multiplier for DC gain approximately in the range 0.001 – 0.02. The multipliers for size and power are in single digits say 1 – 5 and that of the gain bandwidth around 10. To change the range over which the optimization routine explores the solutions, open the required from these three '.m' files. That is, if the multipliers for size and power are to be varied then open meshlinear12.m. In this there are two matrices, xmat and ymat, for c1 and c2 respectively. Change the arrays to the required range. For instance change xmat to [1 2 3 4 5] and ymat to [0.5 1.5 2.5 3.5]. The number of elements in the array do not matter, but it is **necessary** to keep the difference between successive elements to be the same. Similar changes can be carried out in meshlinear.m and meshlinear23.m as needed.

b) Folded Cascode Opamp

The main change in this optimization can be carried out in the fc10per.m. This optimization basically iterates between different initial values of **x**. Hence to change the number of points to be calculated it is necessary to change the upper limit and the step size of the "for loops". This upper limit can be changed by changing the quantity "xupper" in the fc10per file and the step size can be changed by "deltax" variable. Two very important things to be kept in mind are first and foremost, the upper limit is for the design variables. So for the results to be reasonable, it is necessary that this be realistic. Secondly, it is important to realize that there are 7 nested for loops, hence the step size will determine the number of initial guesses along with the upper limit. That is, if the upper limit is 500 and step size is 50, then each for loop will have 10 steps, hence the number of initial guess will be a whopping 10^7 = 10,000,000.

c) Location of PSpice program

For the Two Stage Opamp change the code for "Launch PSpice" button by opening the project in Visual Basic environment and by double clicking on the button in the design mode. Or go to the SpiceOpen_Click() subroutine and change the address in the shell command.

For the folded cascode opamp, change the same piece of code in the PSpiceLaunch_Click() subroutine.

d) Excel files

In case of the two stage opamp, the parameters written in the Excel file "solutions.xls" are $(W/L)_2$, $(W/L)_4$, $(W/L)_5$, $(W/L)_6$, $(W/L)_7$, coupling capacitor Cc and gate bias voltage for M5, $V_{gg,}$ in columns A, B, C, D, E, F and G respectively.

In case of the folded cascode opamp, the parameters written in the Excel file "FCsolutions.xls" are $(W/L)_1$, $(W/L)_3$, $(W/L)_4$, $(W/L)_6$, $(W/L)_8$, $(W/L)_{10}$ and gate bias voltages V_{b1}, V_{b2}, V_{b3} and V_{gg} in columns A through J respectively.

If the Visual Basic program is unable to open Excel file, make sure that the "Excel object" reference is added to the Visual Basic project.