Automatic RFIC Synthesis

A Project report

submitted by

SIKHAKOLLU VENKATA PAVAN SUMANTH

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2022

THESIS CERTIFICATE

This is to certify that the thesis titled **Automatic RFIC synthesis**, submitted by **Sikhakollu Venkata Pavan Sumanth(EE18B064)**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under the supervision of **Dr. Sankaran Aniruddhan**. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Sankaran Aniruddhan
Research Guide
Professor
Dept. of Electrical Engineering
IIT Madras, 600 036

Place: Chennai
Date: 8th June 2022

ACKNOWLEDGEMENTS

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. First of all, I would like to extend my sincere thanks to Dr. Sankaran Aniruddhan for his guidance and constant supervision as well as providing valuable and necessary information about the project.

I would also like to express my gratitude towards Roopesh Pyneni for the co-operation and also the ideas about different algorithms that we used in the project. I offer my sincere thanks to all those who have willingly helped me out completing the project.

ABSTRACT

KEYWORDS: Common Gate LNA, Double Balanced Mixer, Cascaded system of CG LNA and Mixer, Optimization, Spectre, TSMC065

This work mainly focuses on developing a program that automates the design of RF Integrated circuits. Typically, an experienced IC designer would take around a day or two to design a schematic of a particular circuit for the given specifications and in a specific technology. We can reduce this time significantly by automating the design process. This gives much more flexibility in terms of how one wants the design to be; for example, we can design a system whose power consumption is as low or gain as high as possible while still meeting the specifications. Not only this, with the help of this program, the industry can use their human resources and time very efficiently as they can concentrate on very complicated or completely new designs.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
LNA	Low Noise Amplifier
CGA	Common Gate Amplifier
CG LNA	Common Gate Low Noise Amplifier
FoM	Figure of Merit

CHAPTER 1

Introduction

1.1 Problem statement

To design a schematic for given specifications, We need to tune the values of all the circuit components, such as width and length of MOSFETs, resistance, capacitance, and so on. We may also need to tune values like the amplitude of sinusoidal sources. Doing these things manually requires a lot of time and effort. Also, changing from one technology to other will consume some time as one needs to get familiar with new technology before designing the circuits. If we can automate the design flow, we can tackle these issues as computers are much faster than humans. However, doing this is not very straightforward, and hence we may have reliability issues with the results if we irregularly do this. We can improve the chance of getting a working solution by writing the algorithm so that it can replicate what an experienced designer would do if he were the one designing the same circuit.

This work aims to make an algorithm that can automate the design of a schematic and provide an output configuration whose power consumption is as low as possible. The inputs to the algorithm are the circuit topology and the required specifications. The expected output is a valid set of component values for whom the specifications are met and the power consumed is minimum.

1.2 Solution Approach

As mentioned in the introduction, we would like to replicate how an experienced designer goes about designing the same circuit. And that typically would be like,

- Generate some hand calculations for the schematic using circuit laws.
- Using these as initial values, we tune each of them such that the outputs will slowly move towards the specifications. This step is iterative.
- While tuning the component values, one must be careful about the limits of each component value.
- If we can get a satisfactory solution, we will check whether this is fine across temperature and process corners.

Although the algorithm more or less follows the same flow, Implementation is a bit involved. For example, tuning manually requires knowledge about the device. However, to automate this step, we need to develop a quantity that can tell how good the current circuit parameters are. We will look at these details in the following sections.

CHAPTER 2

The Design Flow

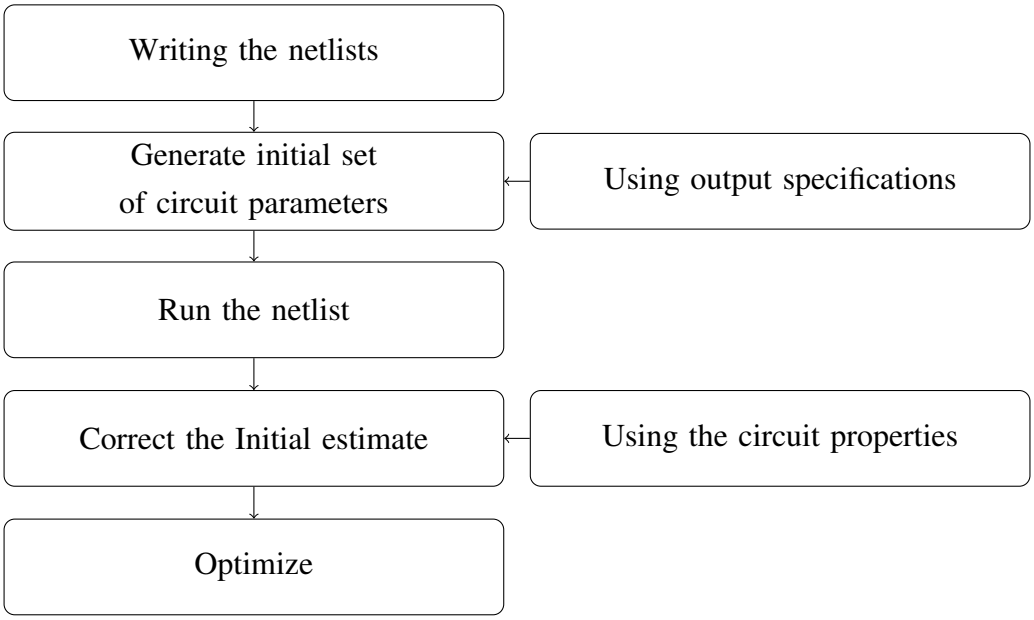

Figure 2.1: Flow of the algorithm

The above flow chart gives a basic outline of how the algorithm works, and we will talk about each step in detail in the following sections.

2.1 Simulator & Netlists

To automate the entire process, we must have complete control over how we change the values of components of the circuit. Hence, we prefer to use netlists to manipulate component values as we can run them from the command line and edit them easily. As we will see in later sections, we would use python for computation, output extraction, optimization, and editing the netlists.

We have used Eldo(Mentor Graphics) and Spectre(Cadence Systems) to simulate the netlists. As Spectre has more robust models for RF designs, we will discuss only Spectre netlists in further sections.

2.1.1 Inside the Netlists...

We define the circuit topology and all components using the netlists. In a netlist, we have variables, model libraries, circuit elements, and analyses like AC, DC, HB, SP, and so on. A sample netlist looks like the following,

```
// Generated for: spectre
// Generated on: May 16 17:20:46 2022
// Design library name: Inverter
// Design cell name: single_inverter
// Design view name: schematic
simulator lang=spectre
parameters fingers=10
include "/cad/library/TSMC/65/gp/oa_pdk/tsmcN65/../models/spectre/crn65gplus_2d5_lk_v1d0.scs" section=tt_rfmos

M0 (OUTPUT INPUT GND GND) nch l=60n w=1.2e-07*fingers m=1 nf=fingers
M1 (OUTPUT INPUT VDD VDD) pch l=60n w=2.4e-07*fingers m=1 nf=fingers

dcOp dc write="spectre.dc" maxiters=150 maxsteps=10000 annotate=status
dcOpInfo info what=oppoint where=rawfile
```

Figure 2.2: A sample netlist of an CMOS Inverter

2.2 Extracting/calculating outputs

Now that we know about netlists, we can build any circuit and run the netlist with suitable analysis to find the required output parameters. However, just by running these analyses, you will not be able to obtain the required quantities. To get the required quantities, one needs to process the output files of these analyses. This section will look at how to get some of these quantities from the raw files.

For a typical RF LNA(Low Noise Amplifier) and a Mixer, the quantities that we are interested in are,

- Gain or Conversion gain(in Mixer's case)
- S11
- Noise Figure
- IIP3 & IIP2(significant for a direct conversion mixer)
- Power
- Bandwidth

Now we will look at how we can obtain these quantities from the raw files.

2.2.1 Gain

There are different ways we can obtain gain at a particular frequency. Depending on how much accuracy one needs, they will choose from AC analysis and HB analysis,

where HB is more accurate than AC, but AC analysis would be sufficient in the case of an LNA. Both of these methods linearize the circuit around the operating point we are interested in, so this number is reasonable for small signals, but if the input signal swing is considerable, then gain will not give a complete picture.

To run AC analysis, we will put an AC source at the input, and in addition, we will give the required frequency range and number of samples in the netlist analysis statement. This generates a file containing small signal voltages at each frequency and node in the circuit. Using this data, we will find the gain as follows,

$$Gain(@f) = \frac{|Voltage\ between\ output\ nodes\ (@f)|}{|Voltage\ between\ input\ nodes\ (@f)|}$$

2.2.2 Conversion Gain

In some circuits like the mixer, the intended output will be at a different frequency than the input excitation. In such cases, we use multi-tone HB analysis and "HBAC" to linearize the circuit around a particular harmonic. In this case, the conversion gain can be obtained as,

$$Conversion\ Gain(f_{in}, f_{out}) = \frac{|Voltage\ between\ output\ nodes\ (@f_{out})|}{|Voltage\ between\ input\ nodes\ (@f_{in})|}$$

2.2.3 S11

S11 describes how good the input matching is, which tells how much power inputted is actually going through the circuit. We can get this value using 'SP' analysis which will directly output a file that contains a matrix of S-parameters where we are interested in S11.

2.2.4 Noise Figure

The noise figure tells us how much noise is getting added by the circuit in addition to the noise that is already present in the source. There are different ways to get noise figures. For example, we can use 'noise' analysis or 'HBnoise' analysis. 'HBnoise' and 'noise' analysis can generate a file with spot noise at any particular frequency. However, spot noise is not enough in circuits like mixers as we need to consider total noise in the whole output band. Hence we use 'Integrated noise figure' as the specification and its

defined as,

$$F_{integrated} = \frac{\int V_{out,n}^2 \cdot df}{\int V_{in,n}^2 \cdot df}$$

Where,

$$V_{out,n}^2 = Noise\ at\ output$$
$$V_{out,n}^2 = Noise\ at\ output\ only\ because\ of\ input\ source$$

The integrated noise is calculated by processing the generated output file with the help of a python script.

2.2.5 IIP3

This specification quantifies how linear the circuit is, but getting this number is not very straightforward as we need to process the node voltages for different input powers to get IIP3. There are two ways to find the IIP3,

- One is from the definition where we input 2-tones whose frequency is off by a small number at the source with the same power/amplitude for each tone. Let the frequencies of the two input tones be f_1 and f_2. To get the IIP3, we need to have the power/amplitude of IM3(Inter Modulation 3) tone, i.e... the tone at $(2 \cdot f_1 - f_2)$ or $(2 \cdot f_2 - f_1)$ at the output. Depending on the circuit, the output power or voltage can be found by doing a 2-tone or 3-tone HB analysis. For circuits like an amplifier, 2-tone HB analysis is enough, whereas for circuits like the mixer, 3-tone HB analysis is required, where the LO signal is the third tone. We repeat this process for different input powers, and finally, we will have the data of the output power of IM3 tone for the given input fundamental tone power. By definition, if we extrapolate the linear region of the IM3 component(ideally has a slope of 3 dB/dB) and the fundamental component(ideally has a slope of 1dB/dB), the input power at the intersection is the IIP3.

  From HB analysis, we will easily get the power of IM3 and the fundamental component. However, for extrapolation, we need a valid region(as the powers must be linear for extrapolating). For that, we consider a fixed number of sample points(Window) and fit the power data to straight lines. This gives us the region's slopes, and we will slide this window through the entire range of input power. Out of all the slopes obtained, we will consider the set whose slopes are close to ideal slopes of $3dB/dB$ for IM_3 and $1dB/dB$ for the $fundamental$. Choosing the region is done by selecting the minimum least square error in slopes which is $(m_{IM_3} - 3)^2 + (m_{fund} - 1)^2$ to get the required region. Here, m_{IM_3}, m_{fund} are slopes of the IM_3 and the fundamental respectively.

- The second method is simple because it uses a closed form expression to find IIP3, which is why this is also not very reliable as the expression needs a good input power which we do not know. However, we can use a decent enough input power to get an IIP3 close to the actual value.

$$IIP3 = P_{in} + 10 \cdot (log(V_{fund}) - log(V_{IM_3}))$$

Where,

P_{in} : Input tone power
V_{fund} : Output fundamental voltage for input power P_{in}
V_{IM3} : Output IM_3 voltage for input power P_{in}

The advantage of the second method is that we only need to run the circuit once(with input power P_{in}), whereas in the first method, we need to run the circuit for different input powers. Hence the first method is too slow but accurate, whereas the second method is fast but not so accurate; hence we can use the second method for most of the optimization, and in the end, we can use the first method to validate our result.

2.3 Hand calculations

Now we have the netlist and a way to find all the required output parameters. However, we still do not have any specific set of circuit parameters that we can use to initialize the algorithm. Certainly, these cannot be random numbers, which might affect how the optimization will end. As we will see later, the optimization algorithm uses the gradient of a specific function to decide what the circuit parameters will take in the next iteration. As this depends on the gradient, this algorithm naturally has the problem of optimizing to a local maxima/minima, and hence the initial set of circuit parameters becomes critical.

We will try to choose the input parameters in such a way that the output parameters are close to the required specifications; hence we need to do this manually for different circuits. In our case, we will make a python script that gives a set of circuit parameters for some given output specifications. These values are obtained by solving the circuit equations as if we would do it by hand. For example, we will use the Shichman-Hodges model for MOSFETs and ordinary equations for linear elements such as resistors, capacitors, and inductors. We must note that, Although these circuit parameters are expected to give the outputs we need, in reality, they will not be close to the outputs at all, and this is because the models that we used to solve the circuit are way simpler than what they were in reality. Nevertheless, these values are good enough for the algorithm's initialization, and with some corrections(using circuit knowledge) to these values, we can start the optimization. We will discuss these in detail in the later section, where we take actual circuits to demonstrate the automation of the schematic design.

2.4 Optimization

The core of this algorithm is optimization, because this step automates the schematic design. In this section, we will see how we find a better set of circuit parameters that satisfy the given output specification from the initial set of circuit parameters.

2.4.1 Cost Function

One important thing to know about a particular set of circuit parameters is how good they are, so we need a way to quantify how well the current circuit parameters are. For this purpose, we are defining the cost function.

Our optimization goal is not to achieve a set of circuit parameters that exactly produce the given output specifications; they can give better outputs than required. Hence, this optimization is not point-based but region-based. For example, say the given specification for gain is 10dB, then our aim is to get a gain better than or equal to 10dB. To take this region based specification into account, we define another function $r(x)$ called the "semi-ramp" function.

$$r(x) = \begin{cases} x & ;\ if\ x \geq 0 \\ 0 & ;\ if\ x < 0 \end{cases}$$

And the cost function is defines as,

$$cost = a_0 \cdot P + \sum_{i=1}^{n} a_i \cdot r((y_i - y_{i,ref}) \cdot d_i)$$

n	:	total number of output specifications
a_i	:	coefficient of i^{th} specification's cost
d_i	:	direction of the specification, $\in \{-1, +1\}$
y_i	:	current value of the i^{th} specification
$y_{i,ref}$	:	reference value for the i^{th} specification
P	:	DC power consumed by the circuit

What is "direction of specification (d_i)"? As we mentioned previously, the given specification is not just a number but a region, though whether the region is above the specification or below the specification must be known to us, and so we pass this information to the cost function as d_i. d_i takes either +1 or -1; for outputs like gain, where a value higher than the specification is a good thing, d_i take -1(minimum specification).

For outputs like noise figure, where a value higher than the specification is a bad thing, d_i takes +1(maximum specification).

2.4.2 Understanding the cost expression

The previous section said that the cost expression would quantify how good the current set of circuit parameters is. To explain this, let us consider the following couple of specifications, a gain of 10dB and a noise figure of 4dB. Assuming $a_0 = 0\ and\ a_1, a_2 = 1$ and for a given set of circuit parameters,

- **Case 1:** The observed gain is around 8dB while noise figure is around 6dB. In this case the cost is 4.
- **Case 2:** The observed gain is around 8dB while noise figure is around 3dB. In this case the cost is 2.
- **Case 3:** The observed gain is around 12dB while noise figure is around 3dB. In this case the cost is 0.

From the above three cases, we can see that the cost is minimal when all the specifications are met, and in such cases, only DC power consumption contributes to the cost. The cost without contribution from power consumption is zero only if all the specifications are met. Hence, our goal for optimization is just minimizing the total cost; in this way, we can get a good solution while reducing power consumption.

2.4.3 Choosing the coefficients in the cost expression

In the cost expression, the only unknown values are the coefficients of each error term(cost due to a particular specification). Although these coefficients are chosen arbitrarily, the value of each coefficient is critical, as they guide the optimization in a particular way. These values,

- Helps us in normalizing all the error terms in the cost expression. The order of each term will be quite different; for example, the specification for gain can be 20dB, whereas noise figure's will be around 2-3dB. If we do not normalize, the algorithm will always prioritize the higher value specification as its error contribution is high.
- Prioritize a particular output specification in the optimization process. Although we normalize to bring every term to similar levels, in some cases, we might need to give some priority to some outputs. In such a case, the corresponding coefficients are relatively higher than others. Without these coefficients, we will not have this control or freedom to prioritize any term.

Keeping the above-mentioned things in mind, We used the following scheme for the coefficients,

$$a_i = \frac{1}{y_{i,ref}}$$

Where,

a_i : coefficient of i^{th} specification's cost

$y_{i,ref}$: reference value for the i^{th} specification

2.4.4 Optimization method

We now have a complete cost expression that suits our needs. We need to opt for an optimization path that does the job for us. Before that, we need to keep in mind that our optimization is of the continuous type where the cost function and the design variables are both continuous functions. In an expansive view, optimization can be of three types,

1. **Random walk**
   - In this sort of optimization, the next set of parameters is completely random. In a way, we will search the entire design space to find the optimum. We need to run a very high number of iterations(maybe 100s of thousands) to get a better result.
   - This kind of optimization will not suit our needs as our functions and variables are continuous. Hence, it is tough to generate random parameters that can cover the entire search space.
   - Next problem is the number of iterations; each iteration takes around 30-50 seconds easily, and hence to run, say even 10,000 iterations, it will take around a week!

2. **Hill climbing(Gradient-based method)**
   - In this method, we will find the gradient of the cost function with respect to the circuit parameters, which gives us a direction for the circuit parameters to move to improve the cost function.
   - This method is efficient in minimizing the cost and would take fewer iterations than a random walk to get a better solution.
   - Only drawback to a gradient-based method is that the optimization settles at a local maxima/minima and will not take us to a global optimum. Hence initialization is critical as to where the optimization finally settles.

3. **Hybrid of Random walk and Hill climbing**
   - Simulated annealing, Genetic algorithms are an example of this method.
   - This kind of optimization has the advantages of both the Random walk and Hill climbing but also has the disadvantage of a high number of iterations and hence not ideal for us.

Out of the above mentioned optimization types, Hill climbing(Gradient based method) is suitable for us. We will discuss this in the next section.

2.4.5 Hill climbing(Gradient based method) update

Although the cost expression involves only the outputs, indirectly, the cost, and directly, the outputs are dependent on the circuit parameters. The gradient of the cost function with respect to the circuit parameters gives us the direction of maximum change. If we follow this direction, we will reach either a local maximum or a local minimum depending on whether we move in the direction of the gradient or the opposite to that.

Our goal is to minimize the cost function, and hence we should move opposite to the gradient direction. The following expression tells how we will update each circuit parameter in each iteration based on the gradient of the cost function with respect to that particular parameter.

$$x_j(k+1) \;=\; x_j(k) - \alpha_j \cdot \frac{\partial cost}{\partial x_j} \cdot x_j^2(k)$$

where,

j	:	index for total number of circuit parameters
x_j	:	j^{th} circuit parameter
k	:	current iteration number
α_j	:	learning factor for j^{th} circuit parameter
$\frac{\partial cost}{\partial x_j}$	:	gradient of cost w.r.t parameter x_j in the $k^{th} iteration$

We update each parameter using the above equation in every iteration to get the next set of parameters. There are a couple of unknowns in the equation, α_j and $\frac{\partial cost}{\partial x_j}$. We will see about these parameters in the following sections. In the equation, we can see that we have multiplied the gradient by x_j^2, this makes sure that the correction term is in the same order as the circuit parameter. This is essential because each circuit parameter differs by many orders; for example, the current will be in mA, the width of MOSFETs will be in μm, whereas resistances may be in the order of kilo-ohms. Hence we use the factor x_j^2 to get the range right.

2.4.6 Finding the gradient of cost function w.r.t circuit parameters

No default function gives the gradient directly; hence we use a python script to calculate the gradient in each iteration. We obtain the derivative using the first principle. We linearize the function by considering a small increment in the input and note the change in the output and the ratio of change in the output to the change in input is the derivative at that point.

$$\frac{\partial\, cost}{\partial x_j} \approx \frac{\Delta cost}{\Delta x_j} \qquad \frac{\Delta x_j}{x_j} = \text{ increment (0.01-0.1\%)}$$

While finding the gradient w.r.t a particular circuit parameter, we only change that parameter by a tiny percentage, say 0.01-0.1%, while keeping all other circuit parameters constant. Now we run the netlist and obtain output parameters from which we can get the information about the cost. In this way, we get the gradient for a particular circuit parameter, and we repeat this for all other circuit parameters as well and can get the complete direction of the gradient.

2.4.7 Choosing the learning factors

The only remaining unknown in the update step is the learning factors for each circuit parameter. This hyper parameter is very critical in determining the pace of the optimization. For example, if the learning factor is too low, then the optimization will be too slow but smooth, whereas if the learning factor is too high, then the optimization will be fast but noisy. By chance, if the learning factor is too high, then instead of optimizing, the cost will blow like a positive feedback loop instead of a negative feedback one. Hence learning factor is very critical in the optimization. We will see some demonstrations in the future sections. Whether the learning factor should be the same for all the parameters is not very clear, and there is no clear evidence to support this claim; hence, for simplicity, we assume the learning factor is the same for all the circuit parameters. Choosing this learning factor is entirely arbitrary; we generally start with 0.1 and see if that works, and if not, we will select one good learning factor with some trial and error.

2.4.8 When to stop the optimization?

Ideally, we should wait until we reach a point where the cost is not decreasing further and flattens with iterations. However, we do not know how many iterations it would take to achieve this condition. Hence, one stopping mechanism is to fix the total number

of iterations and end the optimization after that many iterations. If we want to save time, we can add one more constraint, saying that, after meeting all the specifications, if the power consumption flattens with iterations, we can stop the iteration before the maximum number. However, this may not be reliable as the noise may help us get past the local minimum to achieve an even better minimum. We might miss this if we consider only the slope of power with iterations. Hence more often than not, we run until the maximum iteration.

2.4.9 Optimization summary

1. Choose the coefficients of error terms in the cost expression.

$$cost = a_0 \cdot P + \sum_{i=1}^{n} a_i \cdot r((y_i - y_{i,ref}) \cdot d_i)$$

2. Choose the increment in circuit parameters to find the gradient.

$$\frac{\partial\, cost}{\partial x_j} \approx \frac{\Delta cost}{\Delta x_j} \qquad \frac{\Delta x_j}{x_j} = \text{ increment (0.01-0.1\%)}$$

3. Update the current set of circuit parameters to get better ones.

$$x_j(k+1) \;=\; x_j(k) - \alpha_j \cdot \frac{\partial cost}{\partial x_j} \cdot x_j^2(k)$$

4. Stop the optimization when we hit the iteration limit.

5. **A valid solution :** set of circuit parameters that consumes minimum power while meeting the specifications.

Now that we have a basic understanding of the implementation of the algorithm, we will now consider real circuits to optimize for given specifications. We will go from a simple circuit to the final one by making small changes at a time and also explain any failed attempts to get a better result.

CHAPTER 3

MOS Packages

For our project, we have used 3 different technologies,

- TSMC180
- IBM130
- TSMC65

3.1 MOSFET Characteristics

Parameter	TSMC180	IBM130	TSMC65
Vdd(V)	1.8	1.3	1.0
$L_{min}(nm)$	180	130	60
$\mu_o(cm^2/V)$	273.8	434.7	212.1
$t_{ox}(nm)$	4.1	3.2	2
$C_{ox}(fF/\mu m^2)$	8.42	10.8	17.2
$\mu_n \cdot C_{ox}(\mu m^2/V^2)$	230.5	469.4	364.8
$V_{th,o}(V)$	0.366	0.057	0.317

Table 3.1: MOSFET Characteristics

3.2 MOSFETs usage

We have used TSMC180 and IBM130 when working with Eldo and some early parts of the CG LNA circuit. These are for students to get some design experience and not a robust IC design kit, and hence we moved to TSMC65 in the final stages of building our circuits.

CHAPTER 4

Real(non-ideal) components

Although we have used ideal resistors and capacitors initially, one must use real components that come with the process design kit as they are accurate and close to real-world behavior. For this project, we have used TSMC65 PDK.

4.1 Resistors

4.1.1 Resistor models and Available resistors

There are many resistors available in the TSMC65 package,

S.No	Resistor	Name Full Form
1	rnod	N+ Diffusion Resistor with Salicide
2	rnodwo	N+ Diffusion Resistor without Salicide
3	rnpoly	N+ Poly Resistor with Salicide
4	rnpolywo	N+ Poly Resistor without Salicide
5	rnwod	N Well Resistor under OD
6	rnwsti	N Well Resistor under STI
7	rpod	P+ Diffusion Resistor with Salicide
8	rpodwo	P+ Diffusion Resistor without Salicide
9	rppoly	P+ Poly Resistor with Salicide
10	rppolywo	P+ Poly Resistor without Salicide

Table 4.1: List of available resistors

The above mentioned resistors, each of them belong to one of the following circuit models.

As we can expect, the models that contain diodes are highly non-linear than one with parasitic capacitances. One more thing to note is that in models three and four, the given model is for p-doped resistors, and for n-doped ones, we need to reverse the direction of the diode between nodes and the body. Classification of the resistors by the electrical models is given in the following table.

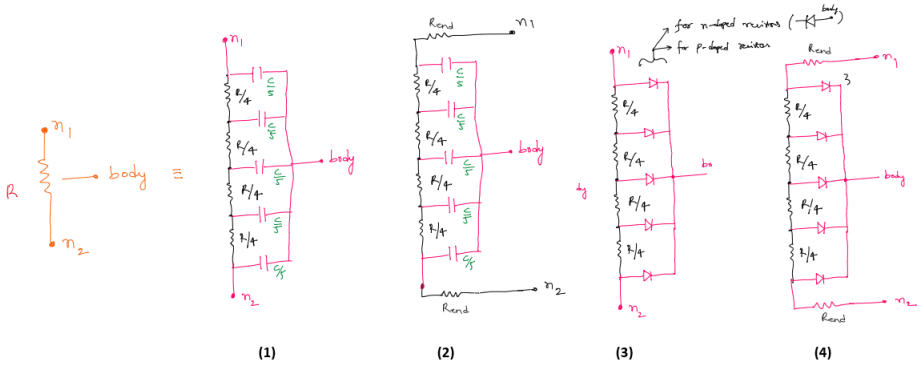

Figure 4.1: Circuit models of the resistors in TSMC65 package

Model number	Resistor
1	rnpoly,rppoly
2	rnpolywo,rppolywo
3	rnwod, rnwsti, rnod, rpod
4	rnodwo, rpodwo

Table 4.2: Classification resistors based on electrical model

4.1.2 Dimensional limits of the resistors

S.No	Resistor Name	Lmin(μm)	Lmax(μm)	Wmin(μm)	Wmax(μm)
1A	rnodl	0.4	100	2	10
1B	rnods	0.4	100	0.4	2
2	rnodwo	0.8	100	0.4	10
3A	rnpolyl	0.4	100	2	10
3B	rnpolys	0.4	100	0.4	2
4	rnpolywo	0.8	100	0.4	10
5	rnwod	9	100	1.8	10
6	rnwsti	9	100	1.8	10
7A	rpodl	0.4	100	2	10
7B	rpods	0.4	100	0.4	2
8	rpodwo	0.8	100	0.4	10
9A	rppolyl	0.4	100	2	10
9B	rppolys	0.4	100	0.4	2
10	rppolywo	0.8	100	0.4	10

Table 4.3: Max and Min dimensions for the resistors

4.1.3 Sheet resistances and Resistance range

S.No	Resistor Name	Rsheet(in Ω)	Rmin(in Ω)	Rmax(in kΩ)
1A	rnodl	15.24	0.61	0.76
1B	rnods	15.24	3.05	3.8
2	rnodwo	104.14	8.3	26
3A	rnpolyl	15.40	0.62	0.77
3B	rnpolys	15.40	3.08	3.8
4	rnpolywo	124.45	10	31
5	rnwod	316	284.4	17
6	rnwsti	605	544.5	34
7A	rpodl	14.55	0.58	0.73
7B	rpods	14.55	2.91	3.6
8	rpodwo	257.3	21	64
9A	rppolyl	14.82	0.59	0.74
9B	rppolys	14.82	2.96	3.7
10	rppolywo	756.2	60	189

Table 4.4: Resistance range and Sheet resistance of the resistors

4.1.4 Process Variation

The following table has details about how sheet resistances change with respect to the process. We need to multiply the multiplier with nominal sheet resistance to get the corresponding sheet resistance for that process.

S.No	Resistor Name	SS Multiplier	FF Multiplier
1	rnod	1.30	0.70
2	rnodwo	1.18	0.82
3	rnpoly	1.30	0.70
4	rnpolywo	1.18	0.83
5	rnwod	1.20	0.80
6	rnwsti	1.20	0.80
7	rpod	1.30	0.70
8	rpodwo	1.16	0.85
9	rppoly	1.30	0.70
10	rppolywo	1.18	0.83

Table 4.5: Process variation of resistors

4.1.5 Temperature Variation

The expression for temperature dependent resistance is as follows,

$$Resistance = R_{sheet} \cdot \frac{length}{width} \cdot (1 + p_{tc1} \cdot (Temp - 25) + p_{tc2} \cdot (Temp - 25)^2)$$

We can notice that the effect of temperature is captured in first-order coefficient p_{tc1} and second-order coefficient p_{tc2} which are listed below

S.No	Resistor Name	p_{tc1} $(*10^{-3})$	p_{tc2} $(*10-3)$
1A	rnodl	2.03	-3.42
1B	rnods	2.31	-4.78
2	rnodwo	1.58	0.42
3A	rnpolyl	2.09	-4.21
3B	rnpolys	2.43	-6.10
4	rnpolywo	0.18	0.25
5	rnwod	2.93	9.99
6	rnwsti	2.30	9.34
7A	rpodl	2.19	-2.37
7B	rpods	2.26	-0.44
8	rpodwo	0.35	1.59
9A	rppolyl	2.30	-3.32
9B	rppolys	2.61	-5.37
10	rppolywo	-0.34	0.73

Table 4.6: Temperature variation of resistors

4.1.6 Observations

Based on the Distortion analysis, frequency analysis, process analysis and temperature analysis the following observations are made,

- **rpodl, rpods, rnpolyl, rnpolys** can't be used in a general cirucuit as their impedence is varying significantly.
- **rnodwo, rppolywo, rpodwo** can't be used, even at frequency of 1GHz, there is significant degradation in the frequency spectrum as the AC resistance is significantly lower than the DC value and phase is greater than 30^o.
- **rnodl, rnods, rnpolyl, rnpolys, rppolyl, rppolys, rnpolywo** can be used in an arbitrary circuit.

4.1.7 Optimal resistor selection

The best choice out of all the available resistors is **rnpolywo** for the following reasons.

- It has low temperature coefficient
- It works fine at frequencies of above 1GHz
- It has a wide range of resistance values: 10Ω to 31kΩ

4.1.8 Resistor parameter selection

We use resistance as the parameter in the optimization but the inputs to the resistor are width and length. Hence we need to get length and width from resistance value.

For rnpolywo the resistance value is,

$$R = R_{sheet} \cdot \frac{L}{W - \Delta W}; \qquad \Delta W = 69.1nm$$

If we fix either length or width we would get other from resistance formula. In our case, we fixed W as W_{min} which is 400nm. From this we get length as,

$$L = (W_{min} - \Delta W) \cdot \frac{R}{R_{sheet}}$$

4.2 Capacitors

There are two available capacitors in the TSMC65 PDK, MIMCAP and MOMCAP. In our project we used only MIMCAP and hence we will discuss about that in this section.

4.2.1 MIMCAP model

The capacitor has 3 terminals; Node1, Node2, and gnode.
Dimenstional limits : $L_{max}, W_{max} = 100\mu m$, $L_{min}, W_{min} = 2\mu m$.

The charge on each plate of the capacitor is modelled as,

$$q = C_{m\_para}[(V_1 - V_2) + vcc_1 \cdot \frac{(V_1 - V_2)^2}{2} + vcc_2 \cdot \frac{(V_1 - V_2)^3}{3}] \cdot C_{mimbb\_corner} \cdot mf$$

and,

$$C_{m\_para} = (1+factmis)[(w_t+mim_a)(l_t+mim_a)10^{12}mim_b+2((w_t+mim_a)+(l_t+mim_a))10^6mim_c]10^{-15}$$

Parameter	Value
w_t	Width of top layer
l_t	Length of top layer
factmis	0
mima	$-0.029 * 10^{-6}$
mimb	1.03
mimc	0.207
tcc1	-$51.1 * 10^{-6}$
tcc2	$-0.03 * 10^{-6}$
vcc1	$36.6 * 10^{-5}$
vcc2	$-18.4 * 10^{-6}$
$cmimbb\_corner$	1
$cm5bb\_mimfac$	1
lay	9

Table 4.7: Parameters for the mimcap model

Along with the main capacitance, there is a parasitic capacitance for Node2 to ground and the value is given as,

$$C_{parasitic} = C_{ox\_para} \cdot \frac{1}{4.357 + 1.42106(lay - 2)} \cdot \frac{cm5bb\_mimfac \cdot mf}{0.0872414}$$

$$C_{ox\_para} = [(w_t \cdot 10^6+0.8)(l_t \cdot 10^6+0.8) \cdot 0.0105357+2 \cdot ((w_t \cdot 10^6+0.8)+(l_t \cdot 10^6+0.8)) \cdot 0.0149] \cdot 10^{-15}$$

4.3 MIMCAP Parameter selection

As we saw the relationship is quite complex but the expression is linear w.r.t mf(multiplier) and hence we would fix W, L at their minimum and find out mf from the required C.

$$mf = \frac{C}{C_{min}}$$

4.4 MOSFET as Capacitor

For MOS capacitor, Capacitance = $C_{ox} \cdot W \cdot L$
$W_{min} = 120nm$,$L_{max} = 20\mu m$ It is also clear that number of fingers is directly proportional to W. and from the simulation the capacitance varies linearly with width if we bias the gate very far from threshold. Hence if we measure capacitance at one particular width then we can naturally extend it using a linear relationship. For this we fix the length at maximum possible value which is L_{max} and the width is given as,

$$W = max(W_o \cdot \frac{C_{req}}{C_o}, W_{min})$$

In the above relationship, C_o is the measured capacitance at a width of W_o

CHAPTER 5

Common Gate Amplifier(CGA)

We will now design a Common Gate Amplifier(CGA) or a Common Gate LNA.

5.1 Schematic

The following is the schematic for the single ended Common gate amplifier.

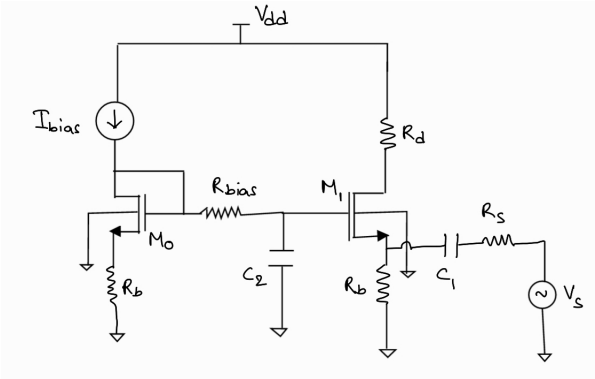

Figure 5.1: Circuit diagram of a single ended CGA

5.2 Specifications

The specifications for the CGA LNA are as follows,

Parameter	Specification	Direction of specification
Gain	10dB	-1
S11	15dB	+1
IIP3	-5dBm	-1
NF	4dB	+1
I_o	Minimize	-
Frequency	1GHz	-

Table 5.1: Output specifications for CG LNA

5.3 Circuit equations

$$
\begin{aligned}
gain &= \frac{1}{2}\, g_m \cdot R_d \\
g_m &= \sqrt{2 \cdot I_{bias} \cdot \mu_n \cdot C_{ox} \cdot \frac{W}{L}} \\
F &= 1 + 4 \cdot \frac{R_s}{R_d} + \frac{R_s}{R_b} + \gamma
\end{aligned}
$$

5.4 Hand Calculations(Automatic initial point)

As mention in the previous section, we hard code these in a python script and hence we need a systematic way of deducing all the circuit parameters from the output specifications.

Design Variables : I_{bias}, W, L, R_{bias}, Rb, Rd, C1, C2.

Algorithm for hand calculations:

1. L = L_{min}(Minimum length of the MOSFET)
2. P1dB$_{min}$ = IIP3$_{min}$-9.6
3. $P_{in} = 10^{-3} \cdot 10^{\frac{P1dB_{min}}{10}}$
4. $V_{osw,min} = gain \cdot \sqrt{8 \cdot R_s \cdot P_{in}}$
5. $V_{osw} = threshold_1 + V_{osw,min}$
6. $I_{bias} = I_{bias,min} = \frac{L^2 \cdot \omega}{6 \cdot \mu_n \cdot R_s} \cdot \sqrt{\frac{1-|S_{11}^2|}{|S_{11}^2|}}$
7. $W_{max} = \frac{3}{L \cdot C_{ox} \cdot \omega \cdot R_s} \cdot \sqrt{\frac{|S_{11}^2|}{1-|S_{11}^2|}}$
8. $R_d = max(\frac{4 \cdot R_S}{Noise\ Factor - \gamma - 1}, 2 \cdot gain \cdot R_s)$
9. $V_{dsat} = 2 \cdot R_s \cdot I_{bias}$
10. $W = \frac{2 \cdot I_{bias} \cdot L}{\mu_n \cdot C_{ox} \cdot V_{dsat}^2}$
11. $R_b = \frac{V_{dd} - V_{osw}}{I_bias} - 2 \cdot R_s - R_d$
12. $C_1 = \frac{threshold_2}{4 \cdot \pi \cdot f \cdot R_s}$
13. $C_2 = threshold_2 \cdot \frac{2}{3} \cdot W \cdot L \cdot C_{ox})$
14. $R_{bias} = max(Rbias_{min}, \frac{threshold_3}{\omega_o \cdot C_2})$

We will get values for all the design variables from the above algorithm, but we need to make sure that all these numbers make sense. For example, all the node voltages must be within V_{dd}, and by chance, if any node's voltage is more than V_{dd}, we need to bring it down before going to the next step. In this particular case, we can see that if we increase I_{bias}, the voltage at all nodes will come down, and we can implement this as a small iterative loop, increasing I_{bias} by 1% each iteration. This will ensure that all the nodes are within V_{dd}.

5.5 Updating the Hand Calculations

As explained in the previous section, although the hand calculated values are calculated from output specifications, we will not get a good result with just these. However, they are good enough to start the algorithm. We need to make some changes to these values to improve the cost.

- Without running the netlist at least once, we will not know the values of V_{th}, C_{gs} of M1 as they are highly non-linear. So we run the netlist with approximated V_{th}, C_{gs}, then we can extract V_{th}, C_{gs} and re-calculate the hand calculations, and these will be slightly better than the previous ones.
- An experienced designer would always check whether the DC current through the transconductor is similar to the bias current. If the DC bias is fine, we will check whether the 'gm' of the transconductor is close to 20mS; if not, we would increase 'gm' by increasing the bias current and correspondingly change the width of the MOSFET. By doing this, we can notice a significant improvement in the outputs.

Algorithm for g_m correction:

1. Increase I_{bias} by 1%. ($I_{bias} \rightarrow 1.01 \cdot I_{bias}$)
2. Change W, R_b accordingly by the relation mentioned in the main algorithm
3. Do this iteratively until we get a 'gm' that is more than 95% of 20mS or for a fixed number of iterations.

5.6 Optimization

In this section, we will see the results of the optimization. However, keep in mind, for this section, the MOSFETs used in the schematics are of TSMC018(Later changed to TSMC065), and everything else, including sources, resistances, capacitors, are ideal elements. In the next section, we will slowly convert everything into a real model schematic.

5.6.1 A Sample Optimization

The following picture depicts the results of a sample optimization,

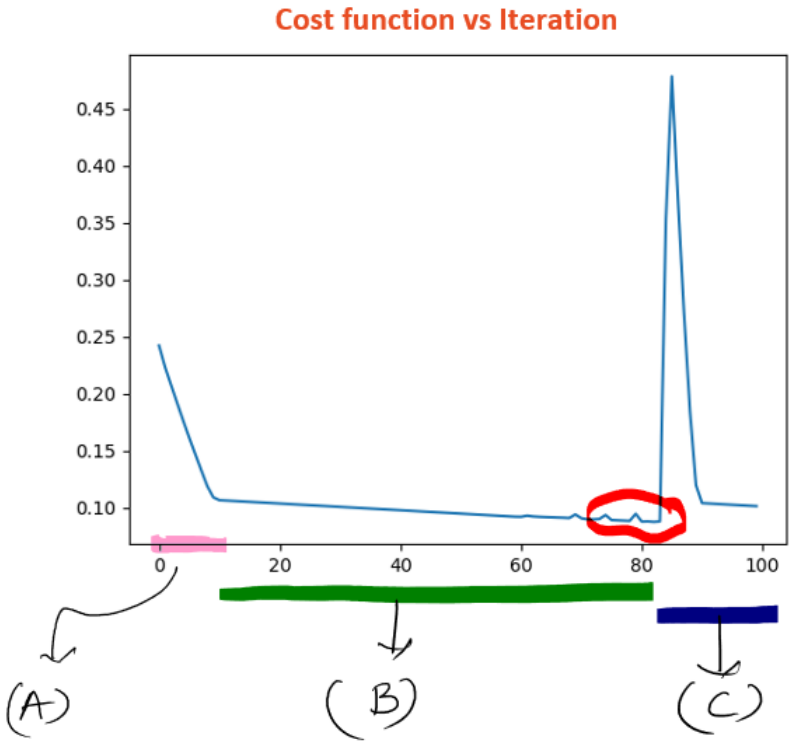

Figure 5.2: Sample Optimization

- **region A:** In this region, the program will try to find circuit parameters that satisfy the given output specifications
- **Region B:** Once we get a valid solution, we will try to maintain the output specifications while reducing DC current consumption. This is the Optimization step, and generally, we would get the best solution at the end of region B. This region would be very noisy as we optimize at the boundary; hence some overflows occur in outputs.
- **Region C:** Once all the output parameters are at the boundary, even for a small perturbation in the circuit parameters, the outputs would get disturbed, and cost would increase suddenly, and in further iterations, the program will try to attain the minimum cost again.

All the optimization need not look like this, they can be of any weird trends but in general the above mentioned regions exist in all kinds of optimizations.

5.6.2 Effect of learning factor on the optimization

In chapter 2, we mentioned that choice of learning factor is critical and we will see that in an actual optimization.

learning factor=0.06

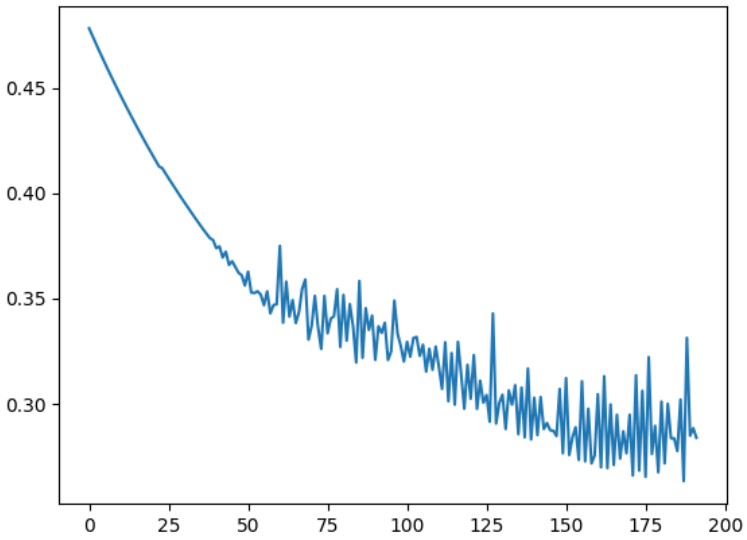

Figure 5.3: Cost vs Iteration for Learning factor of 0.06

$I_{bias}(\mu A)$	592.276
$W(\mu m)$	127.524
$L(nm)$	180
$R_b(\Omega)$	235.877
$R_d(\Omega)$	1.169k
$C1(pF)$	159.155
$C2(pF)$	44.896

Table 5.2: Circuit parameters for $\alpha = 0.06$

$I_o(\mu A)$	592.14
$Gain(dB)$	15.791
$S11(dB)$	-17.092
$IIP3(dBm)$	-4.984
$NF(dB)$	3.991

Table 5.3: Output parameters for $\alpha = 0.06$

learning factor=0.2

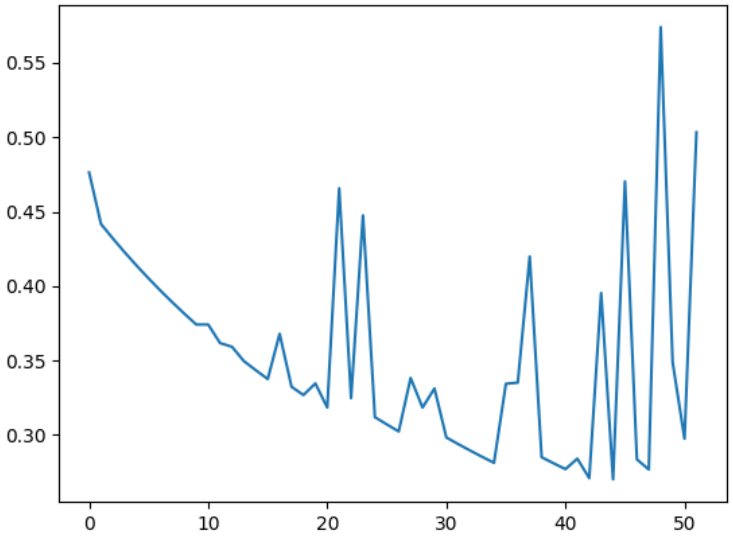

Figure 5.4: Cost vs Iteration for Learning factor of 0.2

$I_{bias}(\mu A)$	608.014
$W(\mu m)$	118.708
$L(nm)$	180
$R_b(\Omega)$	271.712
$R_d(\Omega)$	1.004k
$C1(pF)$	159.155
$C2(pF)$	44.896

Table 5.4: Circuit parameters for $\alpha = 0.2$

$I_o(\mu A)$	608.015
$Gain(dB)$	15.282
$S11(dB)$	-16.812
$IIP3(dBm)$	-4.936
$NF(dB)$	3.927

Table 5.5: Output parameters for $\alpha = 0.2$

Observation: If we carefully look at the final optimized points in both cases, they are very close, but if you look at the cost vs. iteration plot, it took 200 iterations in the

case of $\alpha = 0.06$ whereas it took only 50 in the case of $\alpha = 0.2$. This demonstrates that the speed of optimization depends on the learning factor. One more important observation is that the noisy spikes are very large in the second case, and if the learning factor was even more, higher, that might lead to a blowing up of cost function, and hence learning factor must not be too high nor too low.

5.7 Optimization with Ideal Circuits

We perform optimization of CGA with all ideal components except the MOSFETs and here we compare the results of TSMC018 and TSMC065

Parameter	TSMC018	TSMC065
$I_{bias}(\mu A)$	592.276	638
$W(\mu m)$	127.524	392
$L(nm)$	180	60
$R_b(\Omega)$	235.877	267
$R_d(\Omega)$	1.169k	384
$C1(pF)$	159.155	159
$C2(pF)$	44.896	21.7

Table 5.6: CG LNA Circuit optimzation parameters

Parameter	TSMC018	TSMC065
$I_o(\mu A)$	592.14	720
$Gain(dB)$	15.791	12.3
$S11(dB)$	-17.092	-15.1
$IIP3(dBm)$	-4.984	-3.91
$NF(dB)$	3.991	3.99

Table 5.7: CG LNA Optimized output parameters

5.8 Optimizing using real resistors(from TSMC065)

We perform the optimization by replacing ideal resistors with TSMC065 resistors at room temperature and TT corner. Capacitors are still ideal.

Parameter	Non-Ideal resistors	Ideal resistors
$I_{bias}(\mu A)$	656	638
$W(\mu m)$	335	392
$L(nm)$	60	60
$R_b(\Omega)$	273	267
$R_d(\Omega)$	364	384
$C1(pF)$	159.15	159.15
$C2(pF)$	18.0	21.7

Table 5.8: CG LNA Circuit parameters : real and ideal resistors

Parameter	Non-Ideal resistors	Ideal resistors
$I_o(\mu A)$	723	720
$Gain(dB)$	12.4	12.3
$S11(dB)$	-15.0	-15.1
$IIP3(dBm)$	-3.8	-3.91
$NF(dB)$	3.99	3.99

Table 5.9: CG LNA output parameters: real and ideal resistors

As we can see in case of both Ideal resistors and the Non-ideal resistors, the circuit parameters are very closed to each other and this is because the parasitics did not contribute at 1GHz frequency.

5.9 Optimizing using real capacitors(from TSMC065)

We perform the optimization by replacing ideal capacitors with TSMC065 capacitors at room temperature and TT corner.

We will discuss two cases,

- Both C1,C2 are made using MIMCAP
- Only C1 is made using MIMCAP and C2 is made using a MOSFET. As signal doesnt flow through C2, C2 can be non linear and MOSFETs gate capacitance has highest Capacitance per unit area(C_{ox}) hence we can replace C2 with a gate of MOSFET.

Parameter	Ideal cap	MIMCAP	MOSFET as CAP(C2)
$I_{bias}(\mu A)$	656	822	992.47
$W(\mu m)$	335	205	142.717
$L(nm)$	60	60	60
$R_b(\Omega)$	273	312	347.482
$R_d(\Omega)$	364	293	265.326
$C1(pF)$	159.15	31.8	31.8
$C2(pF)$	18.0	27.5	10.852

Table 5.10: CG LNA Circuit parameters: real and ideal capacitors

Parameter	Ideal cap	MIMCAP	MOSFET as CAP(C2)
$I_o(\mu A)$	723	852	936.42
$Gain(dB)$	12.4	12.0	11.561
$S11(dB)$	-15.0	-15.0	-15.0
$IIP3(dBm)$	-3.8	0.935	2.655
$NF(dB)$	3.99	3.99	3.99

Table 5.11: CG LNA output parameters: real and ideal capacitors

The increase in current from ideal case is attributed to the parasitic cap in MIMCAP which is 10 times smaller than actual cap. Because of this cap, the S11 is degraded and to compensate for S11, The current is increased.

5.10 Optimization of Figure of Merit(FoM)

Figure of Merit: FoM is a metric to tell how good an RF circuit is. Extremely high FoM circuits are of research interest. Hence we can our algorithm to maximize FoM just like how we minimize the cost expression.

$$FoM = \frac{Gain \cdot IIP3(in\ mW)}{(NF-1) \cdot Power(in\ mW)}$$

$$FoM_{dB} = Gain_{dB} + IIP3_{dBm} - (NF-1)_{dB} - Power_{dBm}$$

As we can see there is no S11 term in the expression. But we need to maintain a minimum of -15dB for good input matching. For initialization we use some random specifications and then we maximum FoM.

f (GHz)	$I_d(\mu A)$	Gain(dB)	IIP3(dBm)	NF(dB)	$P(\mu W)$	FoM(dB)
1	101.95	17.65	0.753	9.76	336.94	13.84
2	114.86	16.05	1.624	9.47	381.18	15.91
3	92.5	16.04	1.27	9.94	302.87	17.79
4	89.44	15.84	0.56	9.92	289.19	18.35
5	126.6	14.95	2.14	9.93	413.4	18.47

Table 5.12: IBM130 : Best FoM over multiple frequencies

f (GHz)	$I_d(\mu A)$	Gain(dB)	IIP3(dBm)	NF(dB)	$P(\mu W)$	FoM(dB)
1	136.52	19.027	1.775	9.83	558.6	13.76
2	166.66	19.68	-0.84	9.99	339.13	14.1
3	1284.7	19.56	0.104	4.83	4639.9	14.67
4	1295.5	18.52	0.69	4.96	4682	18.35
5	1139.3	17.83	0.45	5.19	4124	15.49

Table 5.13: TSMC180 : Best FoM over multiple frequencies

If we observe the results, we can see that,

- For high power, noise figure is low
- for low power , noise figure is high

CHAPTER 6

Differential Common Gate Amplifier

We will now convert a single ended CG LNA into a differential CG LNA

6.1 Schematic

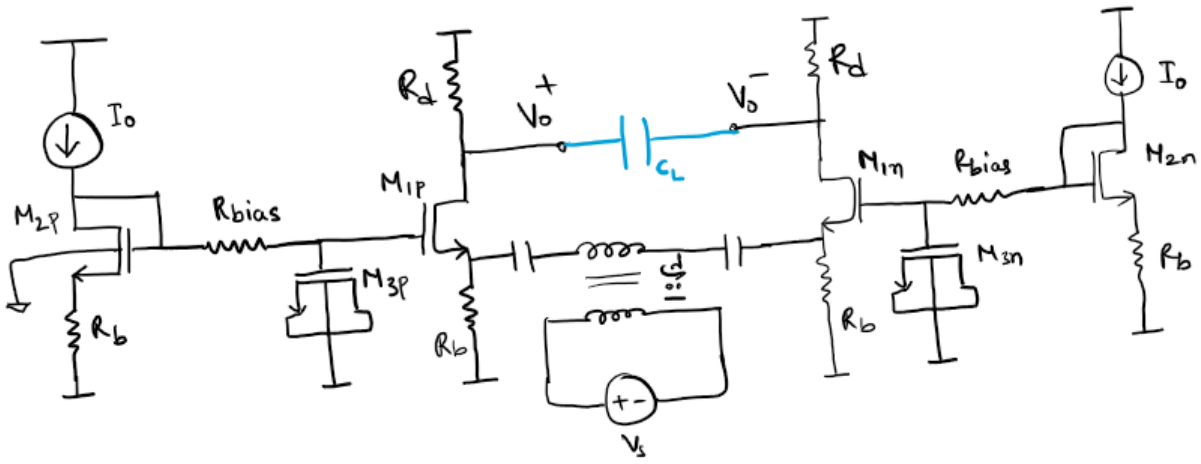

Figure 6.1: Schematic of differential CG LNA

6.2 Specifications

Parameter	Specification	Direction of specification
Gain	10dB	-1
S11	15dB	+1
IIP3	-5dBm	-1
NF	4dB	+1
I_o	Minimize	-

Table 6.1: Specifications for Differential CG LNA

6.3 Comparing Differential and Single ended CGA

The hand calculations part will remain the same for Differential CGA. Resistors are made of rnpolywo, and C1 is made of MIMCAP, while C2 uses a MOSFET. Now for the same circuit parameters(Final optimized point for Single-ended CGA) we will look at output parameters,

Parameter	Values
$I_{bias}(\mu A)$	992.479
$W(\mu m)$	142.717
$L(nm)$	60
$R_b(\Omega)$	347.482
$R_d(\Omega)$	265.326
$C1(pF)$	31.831
$C2(pF)$	10.852

Table 6.2: Circuit parameters for both LNAs

Parameter	Differential CGA	Single ended CGA
$I_o(\mu A)$	936.421	936.421
$Gain(dB)$	11.561	11.561
$S11(dB)$	-15.041	-15.041
$IIP3(dBm)$	5.665	2.655
$NF(dB)$	3.998	3.998

Table 6.3: Output parameters for Differential and Single ended CG LNA

If we look at the output parameters, everything except IIP3 is Identical, and this is no surprise as one should expect of a pseudo-differential amplifier. Just because only half the power flows through each amplifier in the differential case, the IIP3 is improved by 3dB, which is also observed in the output. Hence differential ended CGA is behaving as expected.

6.4 Transforming this into a Wide band LNA

Until now for the gain specification, we considered gain at only given frequency(1GHz) ignoring other frequencies. Now we will make this CG LNA into a wide band one.

6.4.1 Modification to C_1

Everything in the hand calculations remain same except for the bias capacitance at the source as that decides the lower pole. That is modified as follows,

$$C_1 = \frac{1}{2\pi f_{lower} \cdot (2R_s)}$$

Where f_{lower} is the lower 3dB frequency. We do not need threshold in the numerator as we need our pole to be close to f_{lower} not higher than that.

6.4.2 Modification to the cost due to gain

To make sure gain at f_{lower} and f_{upper} is at-most 3dB lower than specification at f_o, we need to modify the gain that is used to find cost as follows,

$$gain = min(gain(@f_{lower}) + 3dB,\ \ gain(@f_o),\ \ gain(@f_{upper}) + 3dB)$$

This makes sure that gain is atleast gain$_{ref}$@f_o and (gain$_{ref}$-3dB)@f_{lower},@f_{upper}.

6.4.3 Optimization

For the following specifications, the optimized results are as follows,

Parameter	Specification	Direction of specification
Gain	10dB	-1
S11	15dB	+1
IIP3	-5dBm	-1
NF	4dB	+1
Frequency	1GHz	-
Freq_lower	50MHz	-
Freq_upper	2GHz	-

Table 6.4: Output specifications wide band LNA

Parameter	Values
$I_{bias}(\mu A)$	1006.2
$W(\mu m)$	159.142
$L(nm)$	60
$R_b(\Omega)$	253.859
$R_d(\Omega)$	253.849
$C1(pF)$	39.222
$C2(pF)$	10.938
$C_{load}(fF)$	50

Table 6.5: Optimized circuit parameters of wide band LNA

Parameter	Differential CGA
$I_o(\mu A)$	950.549
$Gain@f_o(dB)$	11.415
$Gain@f_{lower}(dB)$	7.096
$Gain@f_{upper}(dB)$	11.044
$S11(dB)$	-15.019
$IIP3(dBm)$	5.253
$NF(dB)$	3.999

Table 6.6: Output parameters of optimized wide band LNA

As we can see, gain at all the 3 frequencies are with in specifications and this shows modifying gain for cost works properly. In next section, we use this wide band differential CG LNA with a mixer to form a complete system.

CHAPTER 7

Double Balanced Mixer

Next, we design a Down conversion double-balanced mixer as the next stage to the LNA. This will be a direct conversion mixer, and hence RF and LO frequencies are the same. The only thing is that the RF signal can occupy some band around the main frequency, and we need to design this mixer so that the gain, noise, and linearity across the whole band all satisfy the given specifications.

7.1 Schematic

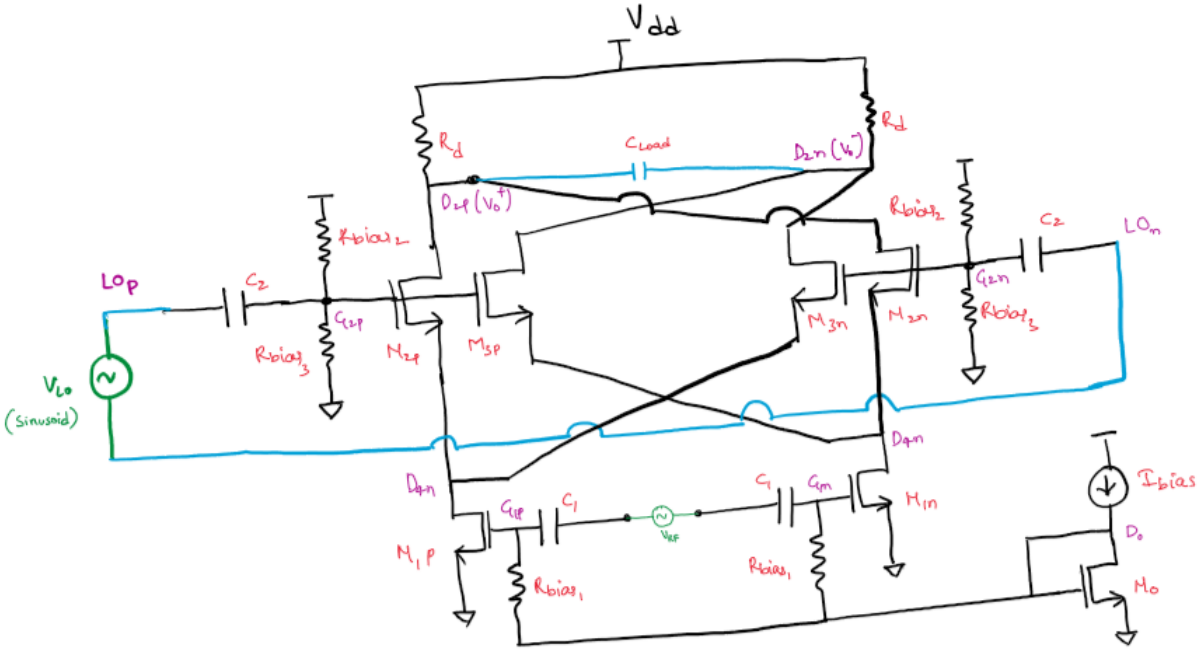

Figure 7.1: Schematic of a Double balanced mixer

7.2 Specifications

The following are the specifications given for the mixer.

Parameter	Specification	Direction of specification
Gain	15dB	-1
IIP3	-10dBm	-1
IIP2	40dBm	-1
Integrated NF	15dB	+1
Bandwidth	10MHz	-1
I_o	Minimize	-
LO frequency	1GHz	-
RF frequency	1GHz+Bandwidth	-

Table 7.1: Specifications for the mixer

For a Direct conversion mixer,

- Frequency of input signal(RF signal) = RF frequency
- Frequency of output signal = (RF frequency - LO frequency)
- Not interested in (RF frequency +LO frequency signal) at the output

7.3 Circuit equations

$$\begin{aligned} \text{conversion gain} &= \frac{2}{\pi}\, g_m \cdot R_d \\ g_m &= \sqrt{2 \cdot I_{bias} \cdot \mu_n \cdot C_{ox} \cdot \frac{W_1}{L_1}} \\ f_{BW} &= \frac{1}{2\pi \cdot R_d \cdot C_{load}} \end{aligned}$$

One problem with finding all the circuit parameters is that it is under-constrained(more variables than equations); hence we use some known methodologies instead of designing for a specific number. We choose a specific Vgs so that g_m is minimum and hence IIP3 is maximized, and we call this Vgs as V_{iip3}. The following plot shows a typical g_{m3} vs V_{gs},

The marked point in the plot is the vgs at which gm3 is close to zero and is ideal for getting a good linearity.

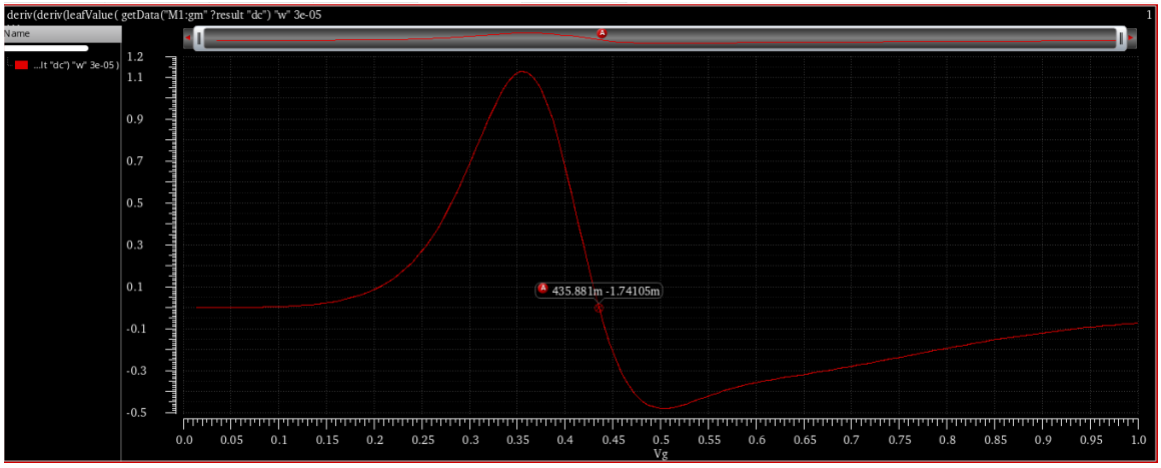

Figure 7.2: gm3 vs Vgs vurve for TSMC065

7.4 Hand Calculations(Automatic initial point)

Design Variables: I_{bias},W_1, W_2, L_1, L_2, R_{bias1}, R_{bias2}, R_{bias3},R_d, C_1, C_2.
Algorithm for hand calculations:

1. $V_{gs1} = V_{iip3}$
2. $V_{th1} = V_{th0}$
3. $V_{th2} = V_{th0} + threshold_1$
4. Relation for Noise Figure, $NF = 1 + \frac{\pi^2}{4 \cdot R_s} \cdot (\frac{\gamma}{g_m} + \frac{2}{g_m^2 \cdot R_d})$
5. $I_{bias} = I_d = \frac{V_{gs1} - V_{th1}}{2} \cdot \frac{\pi^2}{4 \cdot (NF-1) \cdot R_s} \cdot (\gamma + \frac{4}{\pi \cdot gain})$
6. $R_d = \frac{\pi}{4} \cdot (V_{gs1} - V_{th1}) \cdot gain \cdot \frac{1}{I_d}$
7. $L_1, L_2 = L_{min}$
8. $W_1 = \frac{2 \cdot I_d \cdot L_1}{\mu_n \cdot C_{ox} \cdot (V_{gs1} - V_{th1})^2}$
9. $V_{gs2} = \frac{\sqrt{2}}{\sqrt{2} + threshold_2} \cdot (V_{dd} - (1 + \frac{\pi}{4} \cdot gain) \cdot (V_{gs1} - V_{th1})) + V_{th2} - threshold_3$
10. $W_2 = \frac{2 \cdot I_d \cdot L_2}{\mu_n \cdot C_{ox} \cdot (V_{gs2} - V_{th2})^2}$
11. $V_{cm-LO,min} = max(V_{gs1} - V_{th1} + V_{gs3}, V_{gs1})$
12. $V_{diff,min} = 2 * A_{lo}(LO\ Amplitude) = \sqrt{2} \cdot (V_{gs2} - V_{th2}) \cdot threshold_2$
13. $C_1 = threshold_4 \cdot C_{gs1}$
14. $C_2 = threshold_4 \cdot C_{gs2}$
15. $R_{bias1} = \frac{threshold_4}{2\pi \cdot f \cdot C_1}$
16. $R_{bias2} = \frac{threshold_4 \cdot V_{dd}}{2\pi \cdot f \cdot C_2 \cdot V_{cm-LO}}$

17. $R_{bias3} = \frac{V_{cm-LO} \cdot R_{bias2}}{V_{dd} - V_{cm-LO}}$

18. $C_{load} = \frac{1}{2\pi \cdot f_{BW} \cdot R_d}$

Where,

$threshold_1$	:	Correction in threshold
$threshold_2$	:	Multiplication factor for differential signal
$threshold_3$	:	Swing threshold for V_{gs} to maintain saturation
$threshold_4$	:	Threshold for capacitors and resistors

One important aspect is that we need to make sure that A_{lo} is such that the total voltage at the gates of switches would not exceed Vdd. This applies for every iteration too. Hence we limit A_{lo} as follows,

$$A_{lo} = min(A_{lo,calc}, V_{cm-LO}, Vdd - V_{cm-LO})$$

Using the above algorithm, we get all the required circuit parameters and next we need to correct these to get better outputs.

7.5 Updating the Hand Calculations

1. After running the netlist with the hand calculations obtained from above method, we would get an estimate for threshold voltages and C_{gs1}, C_{gs2}. With them we recalculate the circuit parameters.

2. After updating $V_{th1}, V_{th2}, C_{gs1}, C_{gs2}$ we would check whether the transconductor(M1) is in saturation region or not. If M1 is indeed in the saturation region, then we move to the optimization step, and if not, we need to try to correct it and push it towards saturation. The only thing that can push M1 out of saturation is drain voltage being lower than Vgs1-Vth1, which means linear region.

   If M1 is in the linear region, then,
   - We need to make sure that $V_{cm-LO} \geq V_{iip3}$ if not, we cannot bring M1 to saturation.
   - We will increase the widths of LO switches until we push M1 into saturation. For a given gate voltage and current, if we increase the width, then the source voltage would approach $V_g - V_{th}$ and hence if the first condition is satisfied, then we can increase the drain voltage of M1.

7.6 Full optimization of mixer with ideal components

We will use IBM013 MOSFETs and ideal resistors and capacitors.

7.6.1 Automatic Initial Point

The following tables show Initial estimate and corresponding outputs.

Parameter	Values
$I_{bias}(mA)$	17.742
$W1(\mu m)$	40.459
$W2(\mu m)$	15.247
$L1(nm)$	130
$L2(nm)$	130
$R_d(\Omega)$	122.753
$R_{bias1}(\Omega)$	8.33k
$R_{bias2}(\Omega)$	52.256k
$R_{bias3}(\Omega)$	38.333k
$C1(pF)$	3.782
$C2(pF)$	1.425
$C_{load}(pF)$	129.655
$A_{lo}(V)$	0.8

Table 7.2: Automatic initial circuit parameters for the Mixer

Parameter	Values
$I_o(\mu A)$	951.525
$Gain(dB)$	-34.13
$IIP3(dBm)$	10.122
$IIP2(dBm)$	52.8
$NF(dB)$	60.866

Table 7.3: Output parameters for the Mixer

We can see that the outputs are no where near the specifications and they are ridiculously off. And the transconductor operates in linear region. This is the reason we update the hand calculations.

7.6.2 Updated Hand calculations

The following tables show Updated hand calculated circuit parameters and corresponding outputs.

Parameter	Values
$I_{bias}(mA)$	5.419
$W1(\mu m)$	132.475
$W2(\mu m)$	37.383
$L1(nm)$	130
$L2(nm)$	130
$R_d(\Omega)$	122.753
$R_{bias1}(\Omega)$	7.442k
$R_{bias2}(\Omega)$	23.505k
$R_{bias3}(\Omega)$	44.469k
$C1(pF)$	4.235
$C2(pF)$	2.05
$C_{load}(pF)$	129.655
$A_{lo}(mV)$	283.513

Table 7.4: Updated circuit parameters for the Mixer

Parameter	Values
$I_o(\mu A)$	3.933
$Gain(dB)$	9.317
$IIP3(dBm)$	-4.259
$IIP2(dBm)$	47.499
$NF(dB)$	12.215

Table 7.5: Updated output parameters for the Mixer

After updating the parameters, we can see a significant improvement in the output parameters and they are very close to the specifications. And now the transconductor operates in saturation which is desired. But still the current is too high and hence a lot to optimize.

This improvement is attributed to correction in threshold voltage. In the library file, the threshold voltage is around 50mV where as actual threshold voltage is around 400mV. Hence the auto calculations are very off.

7.6.3 Optimization

The following tables show final optimized circuit parameters and corresponding outputs.

Parameter	Values
$I_{bias}(\mu A)$	257.174
$W1(\mu m)$	156.641
$W2(\mu m)$	72.146
$L1(nm)$	130
$L2(nm)$	130
$R_d(\Omega)$	1.507k
$R_{bias1}(\Omega)$	7.562k
$R_{bias2}(\Omega)$	21.505k
$R_{bias3}(\Omega)$	41.469k
$C1(pF)$	4.235
$C2(pF)$	3.25
$C_{load}(pF)$	11.651
$A_{lo}(mV)$	437.449

Table 7.6: Optimized circuit parameters for the Mixer

Parameter	Values
$I_o(\mu A)$	234.426
$Gain(dB)$	15.211
$IIP3(dBm)$	-3.013
$IIP2(dBm)$	58.434
$NF(dB)$	9.892

Table 7.7: Optimized output parameters for the Mixer

As we can see all the outputs met specifications.

7.7 Optimization with TSMC MOSFETs

We will now replace IBM013 MOSFETs with TSMC065 MOSFETs.

7.7.1 Problem faced

After running the optimization, we were not getting a valid solution or in other words the system is not getting optimized. This is because, in case of TSMC065 the V_{iip3} is slightly less than actual threshold voltage and hence the optimization is starting in sub-threshold. After a running many iterations, except gain every other parameter is under specification. Conversion gain is just increasing asymptotically. Hence we are not getting a valid solution.

7.7.2 Modifications in the algorithm to negate the problem

- We will make sure Vgs is above threshold by 100mV. We can do this by increasing I_{bias} iteratively and modifying Rd and W1, W2 correspondingly.
- In the updating step, while updating Rd instead of using bias current, we will use gm(before correction) directly to calculate Rd.

$$\text{Rd} = \frac{\pi}{2 \cdot g_m} \cdot \text{target conversion gain}$$

- With the above two modifications our initial parameters would improve and now to counter the asymptotic settling of gain, we can increase the learning factor in steps so that after many iterations, the change would not be small. The following plot shows a scheme for the learning factor.

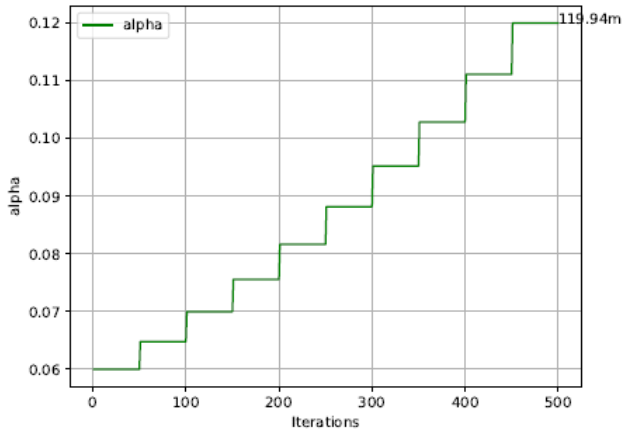

Figure 7.3: learning factor vs Iterations - scheme

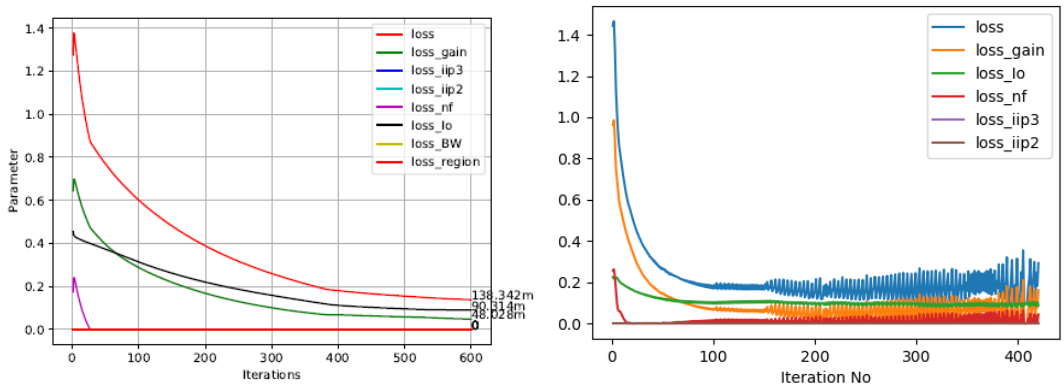

Figure 7.4: Without and with stepping learning factor

As we can see from the plots, After stepping the learning factor, we can get a valid solution from the optimization to the end. The following tables show final optimized circuit parameters and outputs.

Parameter	Values
$I_{bias}(\mu A)$	980.521
$W1(\mu m)$	212.492
$W2(\mu m)$	93.25
$L1(nm)$	60
$L2(nm)$	60
$R_d(\Omega)$	929.853
$R_{bias1}(\Omega)$	12.723k
$R_{bias2}(\Omega)$	263.676
$R_{bias3}(\Omega)$	149.661
$C1(pF)$	2.477
$C2(pF)$	2.465
$C_{load}(pF)$	18.71
$A_{lo}(mV)$	310.967

Table 7.8: Optimized circuit parameters for the Mixer(TSMC065)

Parameter	Values
$I_o(\mu A)$	473.375
$Gain(dB)$	15.045
$IIP3(dBm)$	1.567
$IIP2(dBm)$	58.609
$NF(dB)$	14.972
$Bandwidth(MHz)$	9.999

Table 7.9: Optimized output parameters for the Mixer(TSMC065)

7.8 Optimization with real resistors and capacitors

Now we will replace ideal resistors and capacitors with TSMC065 resistors(rnpolywo) and capacitors(MIMCAP).In the previous section we made certain modifications to get a valid solution but the number of iterations are too high and hence is not good for all specificatios. We will include Length of the transconductor in the optimization loop(previously it was fixed at L_{min}). We are doing this because only W is not sufficient for the certain specifications and hence including Length in the loop gives us one more degree of freedom to get higher gain.

Parameter	Values
$I_{bias}(mA)$	1.158
$W1(\mu m)$	204.433
$W2(\mu m)$	75.09
$L1(nm)$	71.899
$L2(nm)$	60
$R_d(\Omega)$	796.721
$R_{bias1}(\Omega)$	12.65k
$R_{bias2}(\Omega)$	17.834k
$R_{bias3}(\Omega)$	7.646k
$C1(pF)$	2.491
$C2(pF)$	2.54
$C_{load}(pF)$	19.917
$A_{lo}(mV)$	299.636

Table 7.10: Optimized circuit parameters for the Mixer(All real components)

Parameter	Values
$I_o(\mu A)$	234.426
$Gain(dB)$	15.211
$IIP3(dBm)$	-3.013
$IIP2(dBm)$	58.434
$NF(dB)$	9.892
Bandwidth(MHz)	9.999

Table 7.11: Optimized output parameters for the Mixer(All real components)

By including Length in optimization loop we are able to meet all the specifications and hence time. So from now, for mixer we would always include length in the optimization loop.

CHAPTER 8

A cascaded system of CGA-LNA and the Mixer

We have built standalone LNA and a Mixer for given specifications. Now we need to cascade the both to get working LNA-Mixer system.

8.1 Schematic

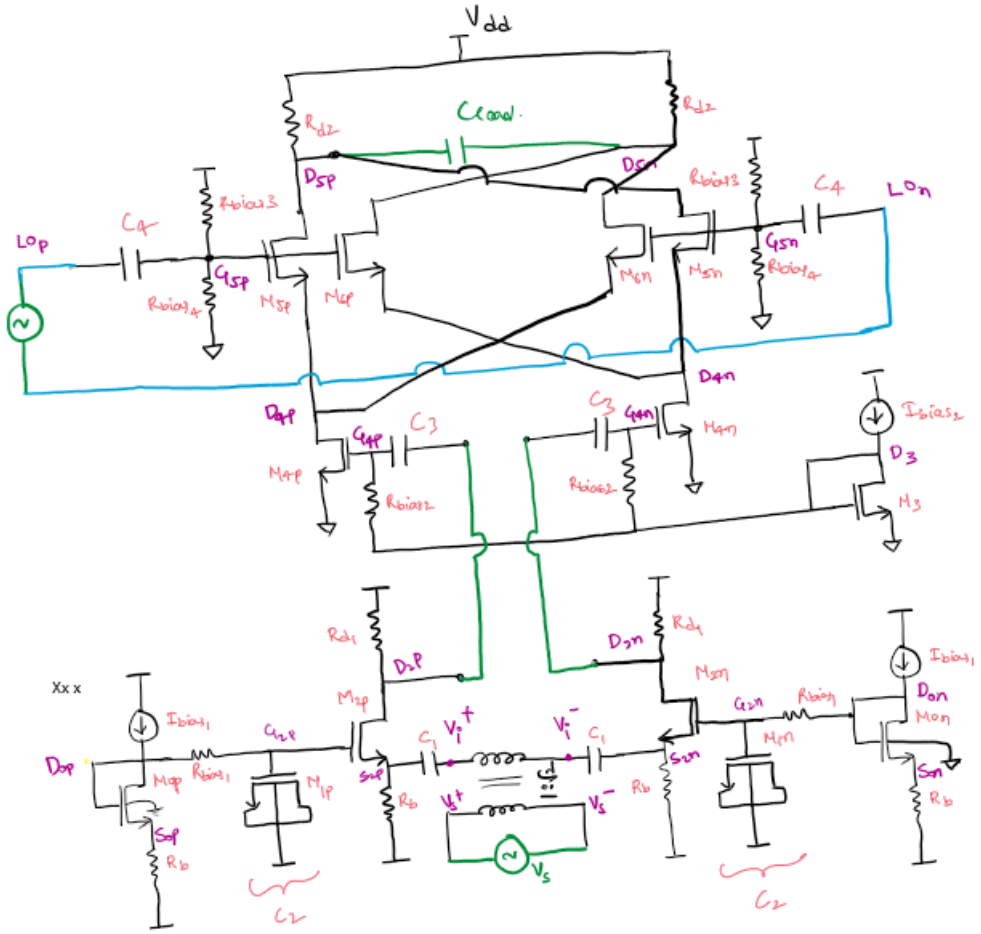

Figure 8.1: LNA-Mixer combined circuit

8.2 Output Specification

The following are the specifications given for the LNA-Mixer system. Note that the specifications are from input of LNA to Output of the Mixer. In the following tables we have listed the outputs that are expected out of the combined system of LNA+Mixer

Parameter	Specification	Direction of specification
Overall Gain	24dB	-1
S11	-15dB	+1
IIP3	-20dBm	-1
Integrated NF	7.5dB	+1
Bandwidth	10MHz	-1
Power	Minimize	-
LO frequency	1GHz	-
RF frequency	1GHz+Bandwidth	-
Freq_lower	50MHz	-
Freq_upper	2GHz	-

Table 8.1: Specifications for the combined LNA+Mixer system

- The combined Noised figure specification is found is using Frii's equation for noise figure,

$$F_{total} = F_1 + \frac{F_2 - 1}{A_1^2}$$

F_1 :Noise figure of first stage alone (LNA)
F_2 :Noise figure of second stage alone (Mixer)
A_1 :Voltage gain of the first stage

- Cascaded IIP3 is found using the relation,

$$\frac{1}{A_{iip3,cascade}^2} = \frac{1}{A_{iip3,1}^2} + \frac{A_1^2}{A_{iip3,2}^2}$$

$A_{iip3,1}$:IIP3 of first stage alone in V (LNA)
$A_{iip3,2}$:IIP3 of second stage alone (Mixer)
A_1 :Voltage gain of the first stage

8.3 Optimization strategies for the combined system

Now that we already written programs to optimize LNA and Mixer alone, we have two ways of optimizing,

- **Optimize LNA and Mixer individually** and cascade them with their best optimized circuit parameters. Later optimize the cascaded system.
- **Optimize LNA+Mixer system from scratch**, build hand calculations, update them and later optimize the circuit.

We will look at both methods in the following sections.

8.4 Method 1: Cascading already optimized LNA, Mixer

For the cascaded system from here on, all the components are non-ideal.

8.4.1 Optimized LNA

We would first optimize LNA with specifications given for it individually. We had already done this before but in this case we need to make sure the load capacitance is at least as large as the next stage's Mixer input capacitance. But we don't know what that value will be and hence we would use a capacitor that is much larger than what it is going to be. Through observations, Mixer's capacitance is generally lower than 40fF and hence we used 100fF single ended or 50fF differential capacitive load at output. And for the following specifications, the optimization results are as follows,

Parameter	Specification	Direction of specification
Gain	10dB	-1
S11	15dB	+1
IIP3	-5dBm	-1
NF	4dB	+1
Frequency	1GHz	-
Freq_lower	50MHz	-
Freq_upper	2GHz	-

Table 8.2: Output specifications LNA alone

Parameter	Values
$I_{bias}(\mu A)$	1006.2
$W(\mu m)$	159.142
$L(nm)$	60
$R_b(\Omega)$	253.859
$R_d(\Omega)$	253.849
$C1(pF)$	39.222
$C2(pF)$	10.938
$C_{load}(fF)$	50

Table 8.3: Optimized circuit parameters of LNA

Parameter	Differential CGA
$I_o(\mu A)$	950.549
$Gain(dB)$	11.415
$S11(dB)$	-15.019
$IIP3(dBm)$	5.253
$NF(dB)$	3.999

Table 8.4: Output parameters of optimized LNA

8.4.2 Optimized Mixer

Simultaneously we can optimize the mixer individually. To make optimization slightly easier, the gain spec has been brought down to 10dB from 15dB. Although for the given individual gain specifications of 10dB each we cannot meet overall gain specification of 24dB and this is intentional as we want to see whether combined circuit can be optimized even if the outputs are off. For the given output specifications, the outputs are as follows,

Parameter	Specification	Direction of specification
Overall Gain	10dB	-1
IIP3	-10dBm	-1
IIP2	40dBm	-1
Integrated NF	15dB	+1
Bandwidth	10MHz	-1
LO frequency	1GHz	-
RF frequency	1GHz+Bandwidth	-

Table 8.5: Specifications for the mixer alone

Parameter	Values
$I_{bias}(mA)$	1.131
$W1(\mu m)$	68.915
$W2(\mu m)$	37.752
$L1(nm)$	62.983
$L2(nm)$	60
$R_d(\Omega)$	312.906
$R_{bias1}(\Omega)$	10.543k
$R_{bias2}(\Omega)$	18.822k
$R_{bias3}(\Omega)$	16.952k
$C1(pF)$	2.989
$C2(pF)$	2.993
$C_{load}(pF)$	12.615
$A_{lo}(mV)$	488.96

Table 8.6: Optimized circuit parameters for the Mixer

Parameter	Values
$I_o(\mu A)$	660.072
$Gain(dB)$	10.392
$IIP3(dBm)$	3.65
$IIP2(dBm)$	66.2
$NF(dB)$	13.59
Bandwidth(MHz)	11.65

Table 8.7: Optimized output parameters for the Mixer

8.4.3 Cascaded system with already optimized circuit parameters

Now that we have optimized points for both LNA & Mixer, we will cascade them to get the outputs of combined system and they are as follows,

Parameter	Values
$I_o(LNA)(\mu A)$	950.55
$I_o(Mixer)(\mu A)$	660.07
$Total\ conversion\ gain(dB)$	21.56
$Gain\ from\ LNA(dB)$	11.47
$S11(dB)$	-15.26
$IIP3(dBm)$	-8.87
$NF(dB)$	7.71
$Bandwidth(MHz)$	11.65

Table 8.8: Optimized output parameters for LNA+Mixer

The following table summarizes the results of independent LNA, Mixer and just the cascade of them(not yet optimized)

Parameter	Independent LNA	Independent Mixer	LNA+Mixer
Gain(dB)	11.415dB	10.392	21.56
S11(dB)	-15.02	–	-15.26
IIP3(dBm)	5.25	3.65	-8.77
Integrated NF(dB)	4.01	13.59	7.71
Bandwidth(MHz)	–	11.65	11.65

Table 8.9: Output summary of LNA, Mixer and their cascade

8.4.4 Optimization of the cascaded system - Method I

We now have a good starting point for the optimization so we can skip hand calculation part and start the algorithm with these set of circuit parameters.

Parameter	Values
$I_{bias1}(mA)$	1.09
$I_{bias2}(mA)$	1.15m
$W1(\mu m)$	161.38
$W2(\mu m)$	77.288
$W3(\mu m)$	37.974
$L1(LNA)(nm)$	60
$L2(Mixer)(nm)$	66.938
$R_{d1}(\Omega)$	280.109
$R_{d2}(\Omega)$	393.169
$R_b(\Omega)$	301.391
$R_{bias1}(\Omega)$	1k
$R_{bias2}(\Omega)$	10.543k
$R_{bias3}(\Omega)$	18.297k
$R_{bias4}(\Omega)$	15.523k
$C1(pF)$	39.22
$C2(pF)$	10.938
$C3(pF)$	2.989
$C4(pF)$	2.993
$C_{load}(pF)$	11.92
$A_{lo}(mV)$	460.387

Table 8.10: Optimized circuit parameters for the LNA+Mixer

$I_{o1}(mA)$	1.028
$I_{o2}(\mu A)$	668.527
$Total\ Gain(dB)$	24.032
$Gain\ from\ LNA(dB)$	12.225
$S11(dB)$	-15.081
$IIP3(dBm)$	-8.82
$NF(dB)$	7.487
Bandwidth(MHz)	11.64

Table 8.11: Optimized output parameters for the LNA+Mixer

As we can see from the results that we were able to meet the specifications even if the initial point is kind of intentionally off and this tells us that this a good way of optimizing the cascaded system. Let us now look at the other methods.

8.5 Method 2: Optimizing the system from scratch

Here, we would optimize the cascaded system from scratch, i.e. generating hand calculations, correcting them, and finally optimizing the system.

8.5.1 Hand calculations

In the previous chapters we have seen hand calculations for independent LNA and Mixer. We will do the same here and the only difference is that instead of capacitor at load we would connect outputs of LNA to inputs of the Mixer.

Parameter	Values
$I_{bias1}(\mu A)$	44.27
$I_{bias2}(mA)$	1.64
$W1(\mu m)$	740.69
$W2(\mu m)$	20.44
$W3(\mu m)$	23.83
$L1(LNA)(nm)$	60
$L2(Mixer)(nm)$	60
$R_{d1}(\Omega)$	316.23
$R_{d2}(\Omega)$	389.86
$R_b(\Omega)$	15.18k
$R_{bias1}(\Omega)$	1k
$R_{bias2}(\Omega)$	11.17k
$R_{bias3}(\Omega)$	116.29k
$R_{bias4}(\Omega)$	23.25k
$C1(pF)$	31.83
$C2(pF)$	102.26
$C3(pF)$	2.82
$C4(pF)$	3.29
$C_{load}(pF)$	40.82
$A_{lo}(mV)$	11.36

Table 8.12: Hand calculated circuit parameters for the LNA+Mixer

Parameter	Values
$I_{o1}(\mu A)$	23.14
$I_{o2}(mA)$	1.0
$Total\ Gain(dB)$	-31.9
$Gain\ from\ LNA(dB)$	-14.97
$S11(dB)$	-0.603
$IIP3(dBm)$	-11.91
$NF(dB)$	55.27
Bandwidth(MHz)	11.65

Table 8.13: Output parameters for the LNA+Mixer

8.5.2 Hand calculations LNA - update

In this step, we will update values of Cgs and Vth of LNA and recalculate hand calculations.

Parameter	Values
$I_{bias1}(\mu A)$	44.27
$I_{bias2}(mA)$	1.64
$W1(\mu m)$	740.69
$W2(\mu m)$	20.44
$W3(\mu m)$	23.83
$L1(LNA)(nm)$	60
$L2(Mixer)(nm)$	60
$R_{d1}(\Omega)$	316.23
$R_{d2}(\Omega)$	389.86
$R_b(\Omega)$	15.18k
$R_{bias1}(\Omega)$	1k
$R_{bias2}(\Omega)$	11.17k
$R_{bias3}(\Omega)$	116.29k
$R_{bias4}(\Omega)$	23.25k
$C1(pF)$	31.83
$C2(pF)$	94.27
$C3(pF)$	2.82
$C4(pF)$	3.29
$C_{load}(pF)$	40.82
$A_{lo}(mV)$	11.36

Table 8.14: Updated circuit parameters for the LNA+Mixer - I

Parameter	Values
$I_{o1}(\mu A)$	23.43
$I_{o2}(mA)$	1.0
$Total\ Gain(dB)$	-31.8
$Gain\ from\ LNA(dB)$	-14.86
$S11(dB)$	-0.610
$IIP3(dBm)$	-11.9
$NF(dB)$	55.62
Bandwidth(MHz)	11.65

Table 8.15: Updated output parameters for the LNA+Mixer - I

8.5.3 Hand calculations Mixer - update

We will update Vth, Cgs of all MOSFETs in mixer and correct the Vdsat of the transconductor if it is in linear region or sub threshold region just like in the previous chapter.

Parameter	Values
$I_{bias1}(\mu A)$	44.27
$I_{bias2}(\mu A)$	90.32
$W1(\mu m)$	740.69
$W2(\mu m)$	998.79
$W3(\mu m)$	1.32
$L1(LNA)(nm)$	60
$L2(Mixer)(nm)$	60
$R_{d1}(\Omega)$	316.23
$R_{d2}(\Omega)$	7.38k
$R_b(\Omega)$	15.18k
$R_{bias1}(\Omega)$	1k
$R_{bias2}(\Omega)$	12.28k
$R_{bias3}(\Omega)$	20.89k
$R_{bias4}(\Omega)$	29.81k
$C1(pF)$	31.83
$C2(pF)$	94.27
$C3(pF)$	2.57
$C4(pF)$	2.57
$C_{load}(pF)$	2.16
$A_{lo}(mV)$	212.13

Table 8.16: Updated circuit parameters for the LNA+Mixer - II

Parameter	Values
$I_{o1}(\mu A)$	23.43
$I_{o2}(\mu A)$	56
$Total\ Gain(dB)$	-8.29
$Gain\ from\ LNA(dB)$	-14.69
$S11(dB)$	-0.610
$IIP3(dBm)$	-9.73
$NF(dB)$	39.75
Bandwidth(MHz)	11.65

Table 8.17: Updated output parameters for the LNA+Mixer - II

8.5.4 Hand calculations LNA - gm update

Finally we will correct the gm of the transconductor of LNA so that it is close to 20mS as we desire. Outputs after the correction are as follows,

Parameter	Values
$I_{bias1}(\mu A)$	1028.2
$I_{bias2}(\mu A)$	90.32
$W1(\mu m)$	115.299
$W2(\mu m)$	998.79
$W3(\mu m)$	1.32
$L1(LNA)(nm)$	60
$L2(Mixer)(nm)$	60
$R_{d1}(\Omega)$	316.228
$R_{d2}(\Omega)$	7.38k
$R_b(\Omega)$	255.427
$R_{bias1}(\Omega)$	1k
$R_{bias2}(\Omega)$	12.28k
$R_{bias3}(\Omega)$	20.89k
$R_{bias4}(\Omega)$	29.81k
$C1(pF)$	31.83
$C2(pF)$	94.37
$C3(pF)$	2.567
$C4(pF)$	2.566
$C_{load}(pF)$	2.156
$A_{lo}(mV)$	212.13

Table 8.18: Updated circuit parameters for the LNA+Mixer - III

Parameter	Values
$I_{o1}(\mu A)$	833.01
$I_{o2}(\mu A)$	56
$Total\ Gain(dB)$	18.541
$Gain\ from\ LNA(dB)$	12.143
$S11(dB)$	-15.343
$IIP3(dBm)$	-7.928
$NF(dB)$	17.734
Bandwidth(MHz)	11.646

Table 8.19: Updated output parameters for the LNA+Mixer - III

Now if we look at the outputs, some of them are within specifications and remaining are close to the required outputs. Hence this way of finding reasonable initial estimate is good. Now with these as initial circuit parameters we will start the optimization.

8.5.5 Optimization of the cascaded system - Method II

With the circuit parameters we got from previous section, We started the optimization algorithm but the results are not good as the cost function was increasing after some iterations, instead of decreasing. Which means that somehow optimization is not working. The following plot shows the abnormal cost function vs iteration number.

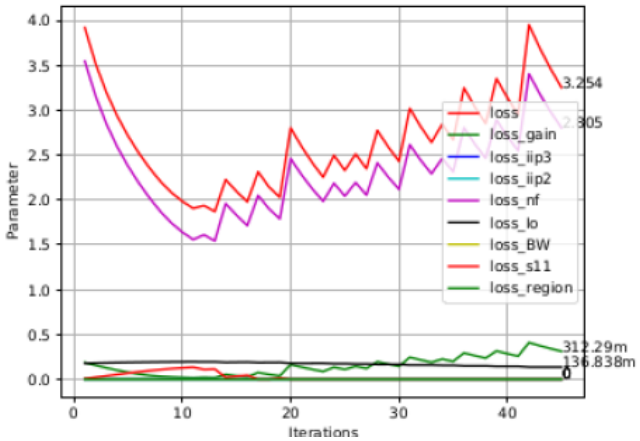

Figure 8.2: Misbehaving cost function vs iteration plot

While debugging, we have found that because of unresponsive Alo(Amplitude of LO Signal), the optimization is not happening. This is happening due to the condition we imposed on Alo,

$$A_{lo} = min(A_{lo,calc}, V_{cm-LO}, Vdd - V_{cm-LO})$$

Optimization wants V_{cm-LO} to increase but this also means that amplitude would reduce which affects switching significantly and hence optimization is going bad. To verify this theory, I intentionally restricted the movement of V_{cm-LO} and then the cost was as follows,

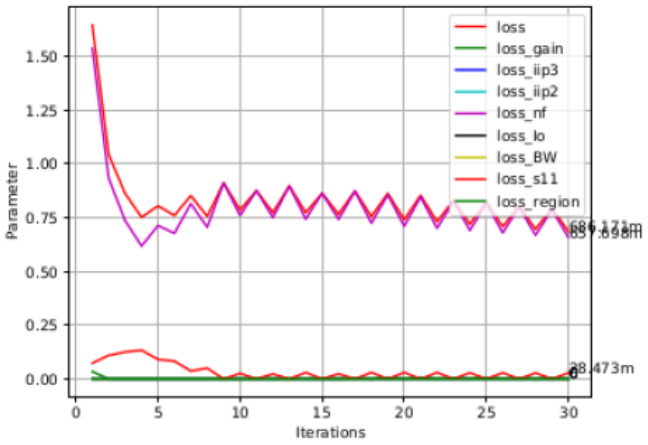

Figure 8.3: Debugged cost vs iterations

We can see from the above plot, the cost is not blowing up and is in fact slowly coming down. This confirms V_{cm-LO} and A_{lo} are indeed the culprits.

Solution: To tackle this we issue, we made a major modification to the circuit schematic. Instead of a sinusoidal source off-setted by V_{cm-LO} as the LO signal, we can use a square wave form switching from 0 to Vdd with realistic rise and fall times. This way we can be assured when it comes to switching and hence ideally the optimization should not have any issues.

8.5.6 Modified schematic with inverters

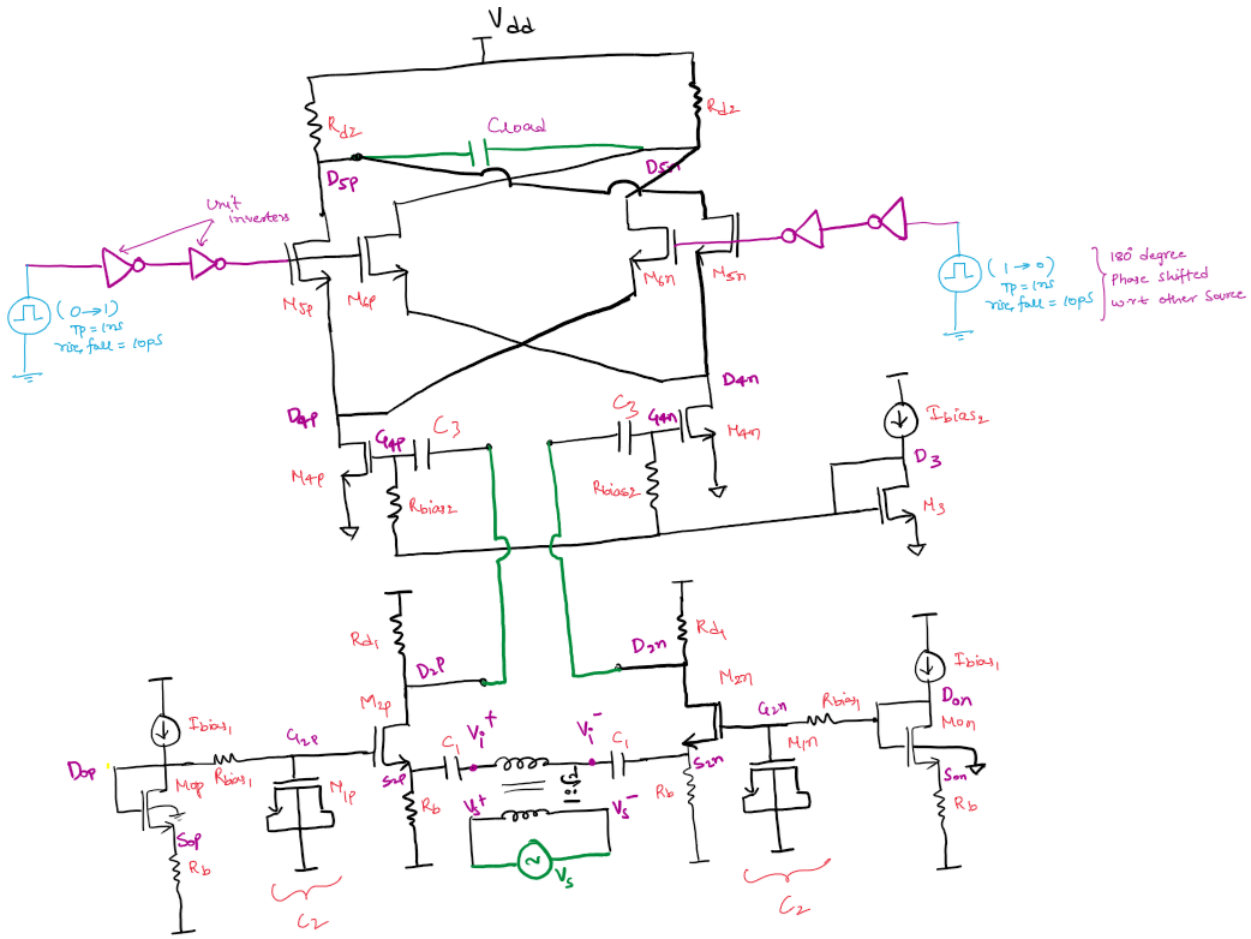

Figure 8.4: Modified LNA+Mixer system with added inverters in LO path

As we can see we have discarded $R_{bias3}, R_{bias3}, C_4, A_{lo}$ from the schematic and hence from the optimization.

8.5.7 Modified Hand calculations

To accommodate the use of a square wave instead of sinusoidal wave, We slightly modify the hand calculations for the mixer.
Design Variables: I_d,W_4, W_5, L_4, L_5, R_{bias2}, C_3.(Only mixer's)

Modified algorithm for mixer hand calculations:

1. $V_{th4} = V_{th0}$
2. $V_{th5} = V_{th0} + threshold_1$
3. To maintain saturation of M5,M6: $I_d \cdot R_d \ = V_{th5} - threshold_2$
4. From mixer gain condition, $V_{gs4} - V_{th4} = min(\frac{4}{pi} \cdot \frac{I_d \cdot R_d}{mixer\ gain}, Vdd - V_{th0})$
5. Relation for Noise Figure, $NF = 1 + \frac{\pi^2}{4 \cdot R_s} \cdot (\frac{\gamma}{g_m} + \frac{2}{g_m^2 \cdot R_d})$
6. $I_d = \frac{V_{gs4} - V_{th4}}{2} \cdot \frac{\pi^2}{4 \cdot (NF-1) \cdot R_s} \cdot (\gamma + \frac{4}{\pi \cdot mixer\_gain})$
7. $R_d = \frac{V_{th3} - threshold_2}{I_d}$
8. $L_4, L_5 = L_{min}$
9. $W_4 = \frac{2 \cdot I_d \cdot L_4}{\mu_n \cdot C_{ox} \cdot (V_{gs4} - V_{th4})^2}$
10. To maintain saturation of M4: $V_{gs5} = Vdd - (V_{gs4} - V_{th4}) - threshold_2$
11. $W_5 = \frac{2 \cdot I_d \cdot L_5}{\mu_n \cdot C_{ox} \cdot (V_{gs5} - V_{th5})^2}$
12. $C_3 = threshold_3 \cdot C_{gs3}$
13. $R_{bias2} = \frac{threshold_3}{2\pi \cdot f \cdot C_3}$
14. $C_{load} = \frac{1}{2\pi \cdot f_{BW} \cdot R_d}$

Where,

$threshold_1$	:	Correction in threshold
$threshold_2$	:	Swing threshold for V_{gs} to maintain saturation
$threshold_3$	:	Threshold for capacitors and resistors

8.5.8 Inverters used in schematic

Inverters are used just to make sure we get realistic square waveform with reasonable rise and fall delays. The inverter used in the schematic is an unit cmos inverter, i.e. minimum possible width combination of,

- PMOS : $W = 4 \cdot \lambda, L = 2 \cdot \lambda$
- NMOS : $W = 2 \cdot \lambda, L = 2 \cdot \lambda$
- In TSMC065 technology, $2 \cdot \lambda = 60nm$

8.5.9 Optimization

With the new hand calculations, we can optimize the circuit. Following tables shows optimized circuit parameters and corresponding outputs.

Parameter	Values
$I_{bias1}(\mu A)$	2091.2
$I_{bias2}(\mu A)$	898.658
$W1(\mu m)$	144.352
$W2(\mu m)$	71.544
$W3(\mu m)$	14.041
$L1(LNA)(nm)$	60
$L2(Mixer)(nm)$	67.196
$R_{d1}(\Omega)$	279.261
$R_{d2}(\Omega)$	587.125
$R_b(\Omega)$	283.764
$R_{bias1}(\Omega)$	1k
$R_{bias2}(\Omega)$	5.392k
$C1(pF)$	31.83
$C2(pF)$	94.37
$C3(pF)$	5.854
$C_{load}(pF)$	2.179

Table 8.20: Updated circuit parameters for the LNA+Mixer - III

Parameter	Values
$I_{o1}(\mu A)$	1206.8
$I_{o2}(\mu A)$	903.987
$Total\ Gain(dB)$	24.785
$Gain\ from\ LNA(dB)$	12.508
$S11(dB)$	-15.261
$IIP3(dBm)$	-10.358
$NF(dB)$	7.5
Bandwidth(MHz)	85.76

Table 8.21: Updated output parameters for the LNA+Mixer - III

From the above results we can clearly see that all the outputs are within specifications implying that square LO signal is working well with the optimization and hence we may even use that for independent mixer.

8.5.10 Comments on both methods

- **Method I:** This method is very reliable as we already have individually optimized circuit parameters. We only need to used them and do a small optimization loop to reach the specifications. Only problem with this method is that we need to work serially as we first optimize individual components and again optimize after cascading, hence takes time.
- **Method II:** This method is very straight forward as we only need to give output specifications and rest is entirely automated hence very easy to use. Only issue with this method is that, compared to method-1, this is slightly unreliable as there is no guarantee of getting a valid solution but this is the case for all optimization hence not an issue.

So depending on whether we already have individually optimized circuit parameters, we either choose method-1 or method-2

CHAPTER 9

Conclusion

9.1 Work done till now

- We have designed an algorithm that can automate the design of a complete schematic given its topology and required output from the circuit.
- We have designed hand calculations and code that optimizes the following circuits,
  - Single ended Common Gate LNA
  - Differential Common Gate LNA
  - Double Balanced Mixer
  - Cascaded system of a CG LNA and the Mixer.
- The code is very flexible to the libraries and the types of components we use. Hence this model is apt for a highly non-linear optimization.

9.2 Future improvements

The work done in this project is primary and there are many things that we can do to take this project forward and couple of them are listed below,

- One of the problem with Hill climbing optimization is that, it settles at a local maxima/minima and will not try to find a global one and this is a fundamental issue and to tackle this we definitely need some randomness that can atleast give a better local optimum. This is exactly what Simulated annealing and Genetic algorithms do. So we can adapt either of them to get better results
- Another interesting direction is to go to next level and automate layouts after the schematics are done. As we know layout design is critical for any schematic to get taped out and see the real world. Hence automating this will significantly cut down the time and effort that goes into design a new chip.