Automatic RF Circuit Synthesis: Power Amplifiers

A Project Report

submitted by

SRIVENKAT A

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 RF Circuit Synthesis: Power Amplifiers**, submitted by **Srivenkat A (EE18B038)**, 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.

Prof. S Aniruddhan
Research Guide
Associate Professor
Dept. of Electrical Engineering
IIT Madras, 600 036

Place: Chennai

Date: June 16, 2022

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude for my research guide, Prof. S Aniruddhan, for spiking my interest in Analog circuit design in my fifth semester and for being a constant source of support throughout the tenure of my B.Tech project. I sincerely thank you for giving me the opportunity to work on this project and for taking time out of your busy schedule to promptly clear my doubts and guide me in the correct direction.

I would like to thank all the professors in IIT-M whose courses have helped me develop strong fundamentals in Electrical Engineering, circuit design in particular and have immensely helped me in my research work.

I would like to thank Pyneni Roopesh (EE18B028) and Pavan Sumanth (EE18B064) who worked with me on the project and laid the foundation for my research work. I would also like to thank my friends and classmates for making my 4 years at IIT Madras some of the most amazing years of my life.

Finally, I would like to thank my parents and sister for their constant encouragement and support in whatever I have pursued in life.

ABSTRACT

KEYWORDS: Circuit Automation; Optimization; Gradient Descent; Analog; Power Amplifier;

The development of EDA's has led to the automation of several stages in the Analog circuit design process. But most of it is post the schematic design process. Analog designers still spend a significant amount of time in parameter tweaking using circuit simulators. We attempt to automate that stage by developing a python code that uses the Gradient Descent algorithm and tries to find the optimal set of circuit component parameters given a specific circuit topology and output constraints. In this project, we choose a Power Amplifier for analysis and try to optimize its component parameters.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
RF	Radio Frequency
IC	Integrated Circuit
PA	Power Amplifier
LF	Loss Function
SS	Small Signal
AC	Alternating Current
DC	Direct Current
HB	Harmonic Balance
RLC	Resistor, Inductor, Capacitor
CSV	Comma Separated Values
AM	Amplitude Modulation
PM	Phase Modulation
op1dB	Output 1dB Compression Point
ip1dB	Input 1dB Compression Point
PDK	Process Design Kit
CLI	Command Line Interface

NOTATION

gdb	Small Signal, dB
I_{ds}	Drain Current, mA
g_m	Transconductance, mS
g_{ds}	Output Conductance, mS
ω_o	Operating frequency, rad/s
μ_n	Electron Mobility, $cm^2/V \cdot s$
α	Learning Rate

CHAPTER 1

INTRODUCTION

1.1 RF Integrated Circuit Design

A typical wireless communication network has a transmitter system and a receiver system. The basic building blocks that make up these RF systems include Low Noise Amplifiers, Direct Conversion Mixers, Power Amplifiers, baseband, passband filters and A/D and D/A convertors. Based on the network specifications, the circuit designers have to ensure that the RF system satisfies multiple constraints on noise, operating voltage swing, power consumption etc.

1.1.1 Design Procedure

The RF system design process generally includes multiple steps:

- Choosing the optimal circuit topology
- Manually calculating the values for all circuit components used based on the output constraints
- Using a RF circuit simulator with real circuit components and simulating the circuit for hand calculated values
- Tweaking the circuit parameters on the simulator to satisfy the output constraints
- Designing the optimal layout for the chosen circuit schematic and then ensuring the output constraints are met on the RC extracted netlist of the layout

1.1.2 Issues with existing solutions

Designing new circuit topologies and analysing these topologies require in-depth knowledge of RF circuit design and are done by experienced analog engineers. But another time-consuming process in the design stage is tweaking the circuit parameters on the simulator to arrive at the optimal set of circuit parameters satisfying all the constraints. It is a highly iterative process, and hence we try to automate it.

1.2 Solving the problem using automation

To converge upon optimal circuit parameters, analog circuit designers slightly change the values of one or more of the circuit parameters at a time, feed it into the simulator and based on the direction in which the output parameters change, update the extent of change in the input circuit parameters. This process can be sped up by algorithmically solving it. That can be done by formulating this as a Regression Inversion (ML) problem or an Optimization problem.

1.2.1 RF Circuit design as a ML problem

If we generate a large but sparse dataset of the output parameters for multiple combinations of input circuit parameters, then we can try to use ML to find the optimal input circuit parameter combination that yields an output value closest to the expected one. The dataset generation will have to be done by running the simulator over changing every circuit parameter by a small fixed amount within a fixed range and recording the output parameters. Once this datacube is generated, we generate a hyperplane over every bin (region where only one circuit parameter changes by one unit), find out which bin the expected output specification falls into and then use regression to find out the optimal set of circuit parameters in that bin from the mathematical function of the hyperplane over that bin.

This approach is a probable long-term goal for the project. But, we haven't yet performed conclusive studies on how optimal the obtained solution for the circuit parameters from the extrapolated hyperplane will be. So, we move forward with the second approach.

1.2.2 RF Circuit Design as an Optimization Problem

Another approach will be to frame a mathematical function that captures how the circuit performs and how much it deviates from the expected performance. This can be done by combining the deviations of separate output parameters from their expected values and using optimization to minimise this deviation. The whole optimization operates as

a closed negative-feedback loop.

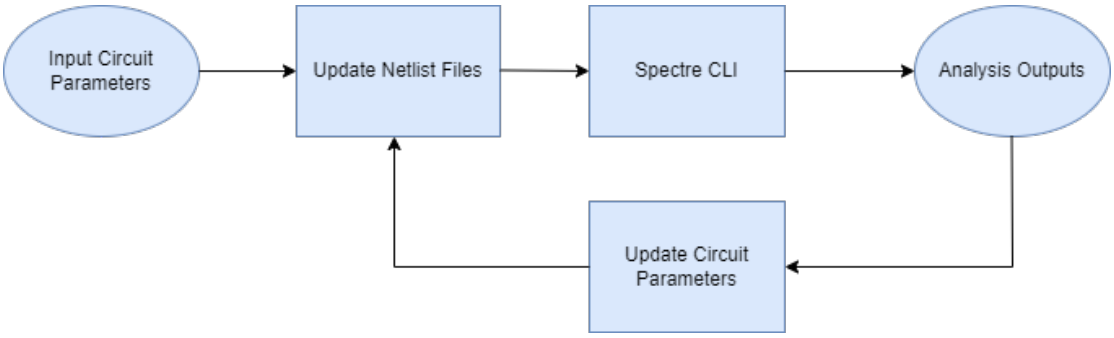

Figure 1.1: Block diagram on closed loop feedback representation of optimization

1.3 Possible levels of Optimization

We can choose to optimize the design process on many levels. A completely automatic circuit design algorithm must be able to choose the optimal circuit topology, the component parameters, the optimal placement and the routing on the layout. This project only addresses the first stage, optimizing for the component values for a fixed circuit topology.

The optimization algorithm is implemented in python and invokes the spectre CLI tool from Cadence to simulate the circuit for the chosen circuit component parameters. We use Gradient Descent to minimise the deviation between observed output parameters and expected output parameters.

1.3.1 The Way Forward

Once the optimization tool can efficiently arrive at the component parameters for a specific building block in the RF chain, we can feed in some commonly used topologies and the tool can be extended to identify the optimal topology by comparing the results obtained from individual optimizations.

Once the schematic stage is optimized, we need to address the issues in optimizing the layout. In RF applications. The effect of parasitics significantly depend on the placement and routing on the layout and it is more complicated to establish a mathematical relation between a layout arrangement and its output performance. Automating the layout for the given schematic is the long-term goal of this project.

CHAPTER 2

POWER AMPLIFIER CIRCUITRY

In this project, we attempt to automate the design of the Power Amplifier used in the TX signal chain. Other blocks like the Double Conversion mixer, LO path circuitry and BB filters are independently optimized, put together and finally optimized together once again to ensure that the combined system has an optimal set of parameters and consumes lowest power possible.

2.1 Class A Power Amplifier

The PA topology we consider in this project is the class A PA. We use a basic nMOS Common Source Amplifier topology and bias it with a nMOS current mirror. We isolate the input voltage source using a VCVS or balun. When cascaded with the previous stage in the TX chain, the mixer, this VCVS can be removed.

2.1.1 Schematic

The schematic is attached in the next page.

2.1.2 Updates to circuit

Then, we extend it to implement a Pseudo-Differential PA and add a differential high-pass matching network to the load side. Using the MN gives us an extra degree of freedom to vary load resistance R_l in the optimisation loop.

Another possible update includes adding a cascode to help with input isolation at the cost of lesser swing limits. But this will reduce the linearity and since we observed that the swing limits were a deciding criterion during optimisation, we didn't continue with the cascoded PA.

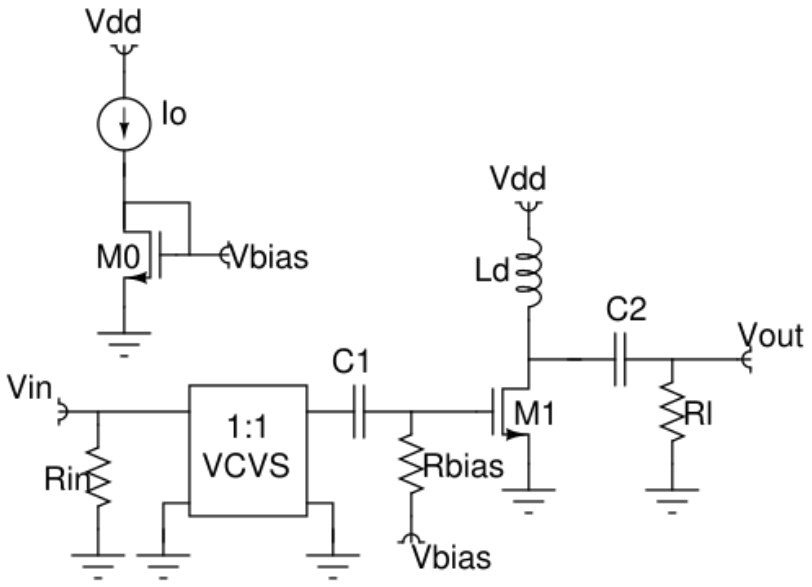

Figure 2.1: Single-ended PA schematic

2.2 Output Specifications

We impose the following constraints on select output parameters of the circuit.

- The operating frequency of the PA is the centre frequency of transmission. That is chosen as 2GHz.
- When the PA is used in a AM-PM modulated system, the phase deviation at the o/p over operating i/p power range has to be minimal. So, the AM-PM deviation should be below 5 deg.
- When the PA is used in AM-AM modulated system, linearity becomes important. The PA should be able to maintain uniform ss gain throughout expected o/p voltage range. So, the output 1dB compression point should be more than 10dBm

Table 2.1: Single-Ended PA Output Constraints

Operating frequency	2GHz
SS Voltage Gain (gdB)	$\geq$ 6dB
AM-PM Deviation	$\leq$ 5 deg
O/P 1dB Compression Point (op1dB)	$\geq$ 10dBm

Table 2.2: Differential PA Output Constraints

Operating frequency	2GHz
SS Voltage Gain (gdB)	$\geq$ 12dB
AM-PM Deviation	$\leq$ 5 deg
O/P 1dB Compression Point (op1dB)	$\geq$ 13dBm

2.3 Design Procedure

We assume all MOSFETs are in saturation and try to find the component parameters for the given set of output specifications. To maximise voltage swings, we assume the nMOS reaches the triode swing limit in the negative peak of o/p voltage swing and reaches the cutoff swing limit in the negative peak of i/p voltage swing.

Because of the drain inductance, the DC value at the drain of the nMOS will be maintained close to V_{dd}. So, the voltage swing at the drain will be roughly between $2 * V_{dd}$ and 0. We need to ensure that this voltage swing is enough to supply expected P_{1dB}.

2.3.1 Hand Calculations

Formulae used to manually calculate the component parameters for the given o/p specifications for a singe-ended PA:

- R_{in}=50ohm (standard)
- $L_0, L_1 = L_{min} = 60nm$ (determined by MOS Technology)
- $V_{power} = 1V$ (determined by MOS Technology)
- $\frac{W_1}{W_0}$ = 1 (width ratio in amplifying circuit to current mirror circuit)
- L_d = 10nH (or largest possible on-chip inductance)
- Taking $V_{outDC} = V_{dd}, P_{out}$ from P_{1dB}, ss gain=G
- $V_{outDC} - V_{outmax} = V_G - V_T + V_{outmax}/G$ (triode limit)
- $V_G = V_T + V_{outmax}/G$ (cutoff limit)
- $V_{outmax} = \frac{V_{dd}}{1+2/G}$ (from swing limits)
- $R_l = \frac{V_{outmax}^2}{2*P_{out}}$ (from P_{1dB})

- $g_m = G/R_l = \mu_n C_{ox}(W/L)(V_G - V_T)$
- $I_d = (1/2)\mu_n C_{ox}(W/L)(V_G - V_T)^2$ (Will drive the current mirror)
- $C_{coup} = \frac{Thresh}{\omega R_l}$ (for AC coupling capacitors)

Changes to be made for a differential PA:

- $R_{l-diff} = R_{l-single}$
- $L_{d-diff-centretapped} = L_{d-single}$

2.4 Matching Network

As discussed in the section Updates to Circuit, we optimize the PA circuit to have a variable R_l and this allows us to sustain the specified op1dB at a lower output swing by using a lower R_l. But the output impedance of the PA should be maintained at 50ohm so that it is matched to the next block on the TX signal chain. So, we design a high-pass wide band Matching Network and connect it to the load side of the existing PA circuitry,

We can choose to use either a T-match or a pi-match MN. Based on the observed range of R_l values and operating frequency range, we observe the pi-match MN has lower inductance values than the T-match equivalent. So, we proceed with that. We also choose to implement the completely differential version of the MN.

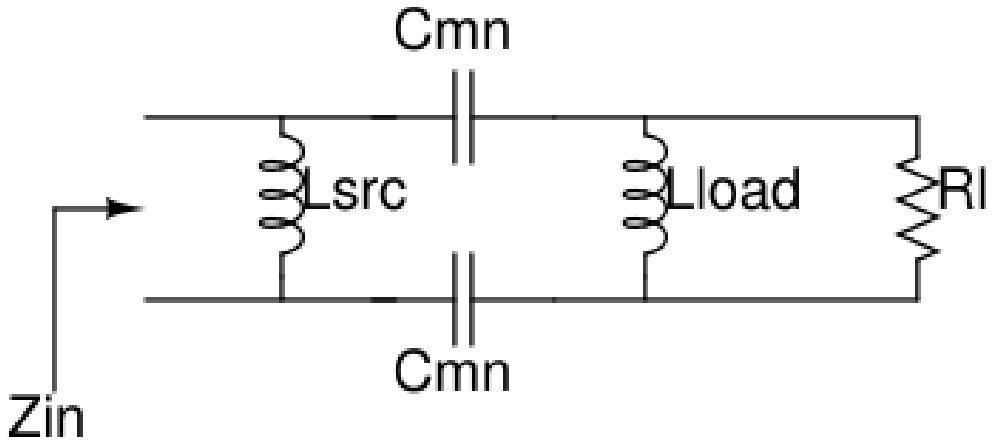

Figure 2.2: Differential MN schematic

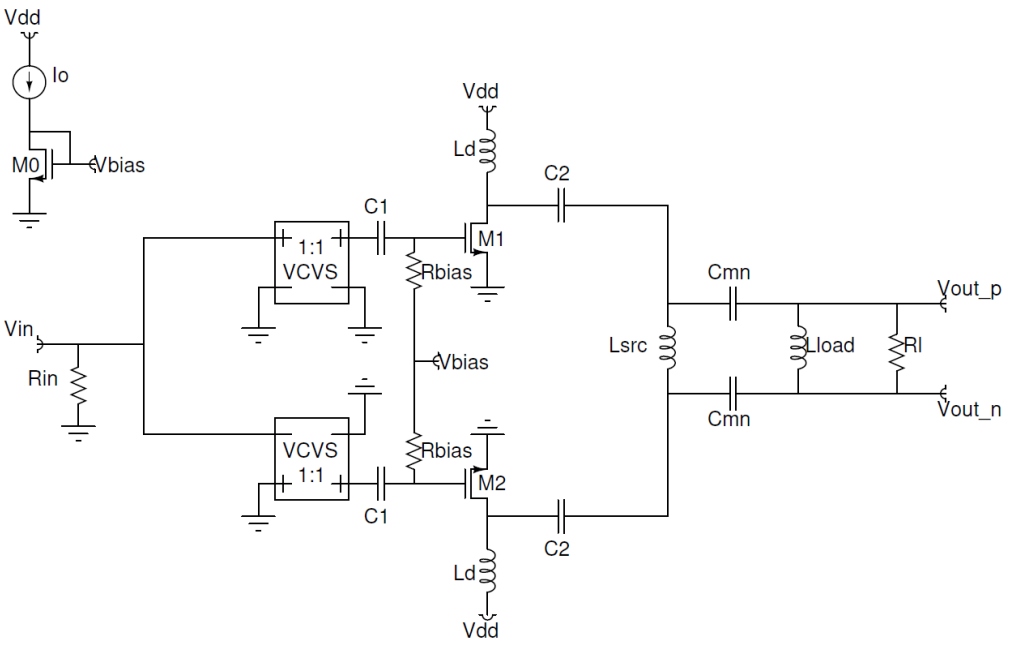

Figure 2.3: Differential PA with MN schematic

2.4.1 Hand Calculation

The MN circuit components to be decided are the source-side inductance, drain-side inductance and the capacitance. The input parameters are R_s, R_l and Q of the MN. We initially choose Q to be fixed at 3 and then update it to choose Q based on the variations in output specs over the specified bandwidth.

Define R_i to be the intermediary resistance.

- R_i = solution of $(Q - \sqrt{\frac{R_l}{R_i} - 1} - \sqrt{\frac{R_s}{R_i} - 1} = 0)$, solved using root-finding.
- $Q_r = \sqrt{\frac{R_l}{R_i} - 1}$
- $Q_l = \sqrt{\frac{R_s}{R_i} - 1}$
- $C_{mn} = \frac{1}{Q\omega_o R_i}$
- $L_{load} = \frac{R_l}{\omega_o Q_r}$
- $L_{src} = \frac{R_s}{\omega_o Q_l}$
- Above or for a single-ended MN. For the differential MN, double the values of L_{src} and L_{load}

CHAPTER 3

GRADIENT DESCENT

Gradient descent is a basic and extremely popular optimization algorithm used in machine learning. It works based on the principle that, when a person standing on top of a hill wants to get to the bottom in the least time, at every step, he chooses the path of steepest descent. This can be used to find the minimum or maximum of a function.

3.1 Optimization using Gradient Descent

This is a first order optimization algorithm because it only operates on the first derivative of the function. So, gradient descent can theoretically be used on any differential function. This algorithm can be used to find the global minima or maxima in convex functions. When the convex function of concern has multiple input variables, we need to use multivariate gradient descent to find the minima (just invert the function in case the maxima is to be found). If the function is not convex, the algorithm might converge on any local minima instead of the expected global one and then choosing the initial point becomes significant.

To tackle a convex optimization problem, we first define a hypothesis function and then a loss function that depends on all the input variables and try to minimise this loss function.

Say θ_{1-n} is the array of n input parameters. h_θ is the hypothesis function.

$$h_\theta = h(\theta_1, \theta_2, ...\theta_n)$$

L_θ is the loss function, defined over the hypothesis function. This is generally chosen to be the mean squared error function between the obtained hypothesis and the expected hypothesis. But we can arbitrarily define this loss function as:

$$L_\theta = L(h(\theta_1, \theta_2, ...\theta_n))$$

Here, L_θ is a multi variable function and hence, the direction of steepest descent for each variable is along its partial derivative. So, in every iteration, we can update all the variables by an amount proportional to the partial derivative of the function w.r.t that variable.

$$\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} L(h(\theta_1, \theta_2, ...\theta_n))$$

Here, α is the learning rate chosen.

With all input variables being controlled independently and having infinite range, they can be simply varied based on their partial derivatives in every iteration. But if the variables have bounds attached to them, we need to implement the constrained gradient descent algorithm.

3.2 Gradient Descent for Circuit Synthesis

We have seen how to frame RF Circuit Synthesis as an optimization problem. In this section, we attempt to implement Gradient Descent to find the optimal set of component parameters for the PA circuit.

To implement gradient descent, we need to define the hypothesis function, the loss function and choose our optimizing parameters.

The hypothesis function must map the circuit component values to the output parameters of concern. So, we need to define a function that takes as input the values for circuit components like resistor, capacitor values and MOSFET widths and calculates the circuit output parameters like op1dB, ss gain and so on. Generating such a mathematical function is complicated. So, we use the simulator to calculate these parameters

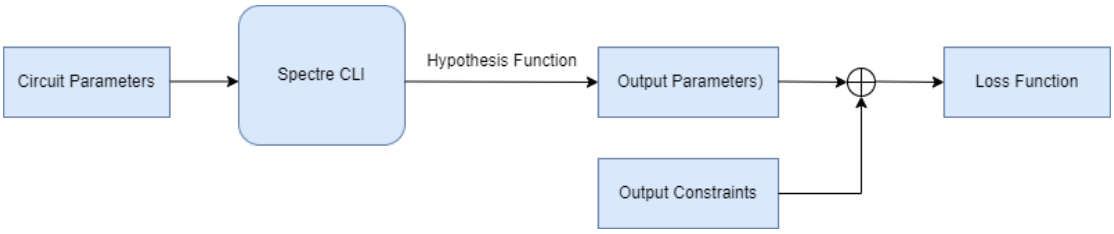

Figure 3.1: Flowchart showing Loss function calc

3.2.1 Defining Loss Function

The loss function must take as input the obtained output parameters of the circuit from the Spectre simulator and compare it with the given set of output specs and estimate the deviation. The loss function should be defined such that minimising this loss function should imply that the circuit is working similar to how we expect it to work.

We independently define loss functions for every output parameter of concern and then combine them to generate a single numerical value of loss for the circuit.

We also define a ramp function that is defined as:

$$ramp(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

We use this ramp function in defining the independent loss functions. This definition only penalizes the circuit if the observed o/p parameter doesn't match the specified limit. This doesn't reward the circuit in any way if the observed gain exceeds the specified limit.

- $Loss_{gain} = A_{gain} * ramp(gain_{spec} - gain_{observed})$
- $Loss_{op1db} = A_{op1db} * ramp(op1db_{spec} - op1db_{observed})$
- $Loss_{am\_pm\_dev} = A_{am\_pm\_dev} * ramp(am\_pm\_dev_{observed} - am\_pm\_dev_{spec})$
- $Loss_{I_o} = A_{I_o} * I_{o-observed}$
- $Loss_{p\_higher\_harms} = A_{p\_higher\_harms} * ramp(p\_harm\_ratio_{spec} - p\_harm\_ratio_{observed})$

The $Loss_{p\_higher\_harms}$ term is added to penalise the circuit for distributing power in higher harmonics. This helps to improve linearity of the circuit. We impose the constraint that the power in higher harmonics should be atleast 15dB lower than the power in fundamental harmonic.

There are multiple ways to combine the independent loss functions. We go with the simple approach of linearly combining them to get net loss.

$$Loss = Loss_{gain} + Loss_{op1db} + Loss_{am\_pm\_dev} + Loss_{I_o} + Loss_{p\_higher\_harms}$$

The variables $A_{gain}, A_{op1db}, A_{am\_pm\_dev}, A_{I_o}, A_{p\_higher\_harms}$ are the loss weights that allow us to prioritise the optimization of one output parameter over another. Having a large loss weight would mean it would have a more significant contribution in the overall loss function and hence the gradient descent algorithm will be more focused on optimizing that parameter.

We don't want to prioritize one output parameter over another and need the optimizer to equally focus on optimizing all output parameters while reducing the bias current. Since different output parameters have different units and their values in different ranges, we normalise them using the loss weights. We choose the loss weights to be reciprocals of the specified value.

Table 3.1: Loss Weights for Optimization

$Loss_{gain}$	1/6
$Loss_{op1db}$	1/10.0
$Loss_{am\_pm\_dev}$	1/5.0
$Loss_{I_o}$	1000
$Loss_{p\_higher\_harms}$	1/15

3.2.2 Fixing Learning rate

The parameter learning rate controls the extent by which each variable is updated in a single iteration. Choosing a small learning rate would mean the algorithm takes a large time to converge while choosing a large learning rate might lead to instability. So, we try to implement alpha-rate decay where we reduce the learning rate with iterations to ensure convergence in minimum iterations. Learning rate is denoted by α.

Different α parameters can be chosen for different input variables. But we keep the learning rate uniform for all the optimizing input variables at the start. In further modifications, we revisit to using different learning rates for those variables that we don't want to change much during optimization.

We can fix the bounds for α and vary it linearly or logarithmically over every iteration. We can also fix a multiplicative factor (typically <1) and vary α exponentially over a iteration after checking if the system approaches instability.

Linear-sweep: $\alpha = \alpha_{start} + ((\alpha_{end} - \alpha_{start}) * (iter + 1)/n\_iter)$

Logarithmic-sweep: $\alpha_{log} = \alpha_{start-log} + ((\alpha_{end-log} - \alpha_{start-log}) * (iter + 1)/n\_iter)$

Normal: $\alpha = \alpha * \alpha_{mult}$ if $loss_{iter+1} > loss_{iter}$

3.2.3 Choosing Optimization Parameters

We categorise the circuit component parameters into independent and dependent parameters. Some of the independent parameters are fixed to constant values like fixing R_{in} to 50ohm. The dependent parameters include choosing the coupling capacitor values based on a threshold and the associated resistor value in the AC coupler.

So, the final circuit parameters chosen for optimization are: $R_{bias}, L_d, I_o, W, R_l$. When we include the matching network, we also include the components in the MN in the optimization loop. So, $L_{src}, L_{load}, C_{mn}$ also get optimized.

CHAPTER 4

TSMC65 TECHNOLOGY & PDK

All the experiments and analysis in this project have been done on TSMC's 65nm technology node.

4.1 MOS Characteristics

We only use nMOSFETs in our PA schematic. The technology node determines determines multiple parameters like the supply voltage and min. channel length. The MOSFET models we use are n-channel ones denoted by "nch". We start by designing the circuit with ideal components and progressively replace them with real components. Passive components like resistors, capacitors and inductors are specifically designed for use in the RF domain. The parasitics of these components are modeled in the TSMC 65nm PDK.

Table 4.1: TSMC 65nm MOS Characteristics

Parameter	**Value**
V_{dd}	1.0 V
L_{min}	60 nm
μ_n	212.1 $(cm^2/V \cdot s)$
C_{ox}	17.2 $(fF/\mu m^2)$
$\mu_n C_{ox}$	364.8 $(\mu A/V^2)$
t_{ox}	2 nm
V_{th_0}	0.317 V

4.2 Non-Ideal Resistors

Using sheet resistances to implement resistors, we initially fix the length and width to L_{min} and W_{min} respectively to find R_{ref} and fix other parameters w.r.t this R_{ref}.

$$R_{ref} = R_{sheet} * \frac{L}{W}$$

If $R < R_{ref}$,

$$L = L_{min}, \quad W = L_{min} * \frac{R_{sheet}}{R_{expected}}$$

If $R > R_{ref}$,

$$W = W_{min}, \quad L = W_{min} * \frac{R_{expected}}{R_{sheet}}$$

The TSMC65 PDK has multiple resistor models depending on the structure, doping and presence or absence of Salicide contacts. We only analyse them from the top level in terms of their available resistance range, distortion, temperature coeff. and process variations. We choose the "rnpolywo" resistor that is a N+ doped polysilicon resistor without Salicide. We choose this because it is available over a wide range of values and has low variations over temperature and process corners.

Specs of rnpolywo:

- $R_{min} = 10\Omega$, $R_{max} = 31k\Omega$
- $R_{sheet} = 124.45\Omega$
- $L_{min} = 0.8\mu m, L_{max} = 100\mu m, W_{min} = 0.4\mu m, W_{max} = 10\mu m$
- 1st order temperature coefficient = 0.18 * 10^{-3}

After simulations, we observe that the AC resistances obtained from the above calculation method are 1.3x the expected value. So, for the original hand calculations, we use this approach and then let the optimization loop set the lengths and widths of the MOS resistors directly.

4.3 Non-Ideal Capacitors

We can implement capacitors using either MIMCAP or MOSCAP models from the TSMC65 PDK. For capacitors in the path of the signal, we need to use MIMCAPs over MOSCAPs as they are more linear and less sensitive to process and temp. variations. In the PA circuit, capacitors are only used for AC coupling and in the MN, both placed in the signal path. So, we only use MIMCAPs in the circuit.

To size the MIMCAPs, we calculate the capacitance values for the largest and smallest possible dimensions and then vary the multiplicative factor to get specific capacitance values.

Largest possible value for width, length is $100\mu m$ and observed capacitance is $10pF$. Smallest possible value for width, length is $2\mu m$ and observed capacitance is $5.64fF$.

If C > 10pF:

$$mf = 1 + int(\frac{C}{10 * 10^{-12}}), \quad width, length = W_{max} = 100\mu m$$

Else:

$$mf = 1 + int(\frac{C}{5.64 * 10^{-15}}), \quad width, length = W_{min} = 2\mu m$$

Since the exact value of the coupling capacitors don't significantly affect the output parameters, we don't independently optimize them. We fix them based on the value of the associated resistor just ensuring that the cutoff frequency is much lower than the operating frequency (by a user-defined threshold).

4.4 Non-Ideal Inductors

The TSMC65 PDK has spiral inductors that are more complicated to model and have more design parameters than the resistors and capacitors. We need to set the values for width, radius, number of turns, gdis and spacing for the traces. When designing for a single case, spectre offers an option of using an Inductor finder where it has a densely populated table of inductance values for multiple combinations of the 5 input parameters. We need to enter the inductance and the quality factor to find the nearest available combination.

We couldn't replicate the inbuilt spectre finder and as the optimization algorithm needs to call the function multiple times to calculate inductor parameters for specific L, Q values, we attempt to create a customised inductor finder. We manually sweep all the five parameters over a predetermined range and store the table of inductance and quality factor values. Then based on the specified L, Q values, we find the nearest L, Q combo and the corresponding input parameters. The following analysis is done for the "spiral_std_mu_z" inductor.

Table 4.2: TSMC 65nm Inductor Sweep Parameters

Parameter	**Range**	**# Points**
Width	$3\mu m - 30\mu m$	10
Radius	$15\mu m - 90\mu m$	10
Number of turns	0.5 - 5.5	11
Gdis	$10\mu m - 50\mu m$	5
Spacing	$2\mu m - 4\mu m$	3

The sweep is done at a center frequency of 1GHz and the data is written into a CSV file. To find the closest combination of L ad Q values, more emphasis is given to matching the L value. So, we first filter out the combinations that give an inductance value between 0.98*L and 1.02*L, i.e., all combinations with inductance value within 2% of the expected value.

Among these entries, we find the combination with the lowest normalised mean sq. error in L, Q values. If we want to find the optimal input parameters for the specified L, Q, we minimise $(\frac{l-L}{L})^2 + (\frac{q-Q}{Q})^2$ to get the closest l,q and choose its input parameters as the optimum set.

CHAPTER 5

OPTIMIZATION ALGORITHM

Here, we discuss the different stages in the optimization algorithm and the design choices made at each step. The entire algorithm runs on a python back-end and invokes Spectre-CLI multiple times every iteration to run simulations over a modified set of input parameters. It extracts the output parameters from the resultant log files of the spectre run and processes them to identify how to update the input parameters in the next iteration. The entire algorithm runs in 2 distinct stages: Pre-Optimization and Main Optimization.

5.1 Pre-Optimization Stage

5.1.1 Initial Point Selection

Since the loss function chosen by us for gradient descent is not a convex function, there is a chance of the algorithm converging at a local minima instead of the global one. So, we need to choose the initial condition carefully. The algorithm has 2 options: either to manually choose the circuit parameters at the start of the optimization loop or let the code calculate the initial parameters automatically.

Automatic Initial Point

The hand calculation steps detailed here can be used to fix the component parameters based on the output specs given. The python code automatically calculates the component values and passes them to the optimization loop.

Manual Initial Point

We encountered cases where the optimization would stop in the middle due to an exceptional case or where we needed to experiment by varying just one of the circuit

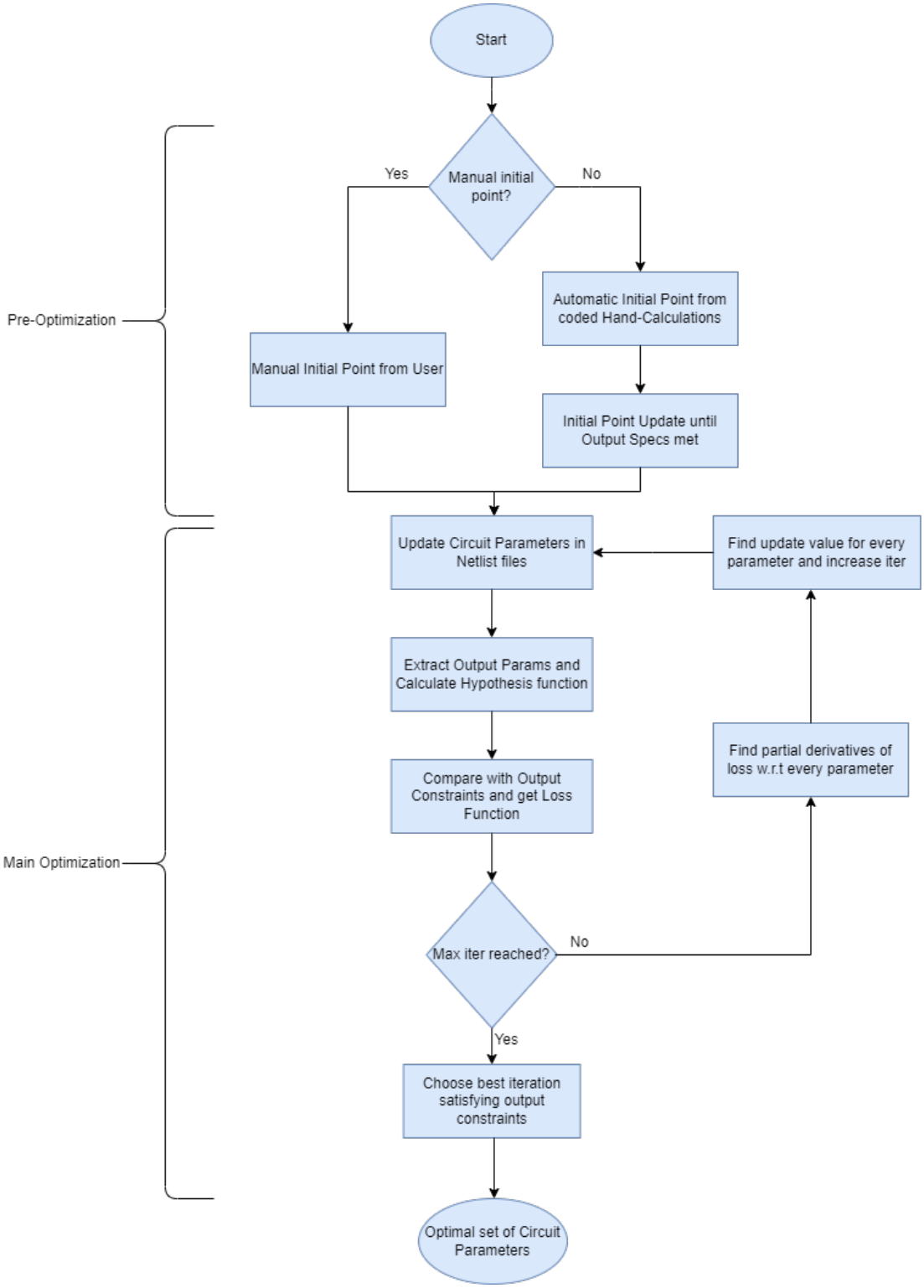

Figure 5.1: Flowchart of Overall Optimization

parameters to see if the optimization converged at a different point. To facilitate such cases, the algorithm also supports an option to bypass the automatic hand calculations and we can manually enter the circuit component parameters at the start of the optimization loop.

5.1.2 Initial Point Update

This stage is triggered only when the initial point is chosen automatically. We observed that despite calculating the circuit parameters based on the given output specs, the hand calculations assume an ideal circuit. So, when we simulate the circuit with the calculated parameters, the output specs are not satisfied, most notably the op1db spec. So, we keep increasing the MOSFET width while keeping other parameters constant until the output specs are satisfied.

This stage also tackles with the issue of the multiplicative offset in non-ideal resistors. While R_l is left to be optimized in the main optimization stage, R_{in} is fixed and doesn't need to be optimized depending on the output specs. So, we fix it in the pre-optimization stage itself.

Let R_{in-exp} be the expected value of R_{in}, R_{in-ext} be the extracted value of R_{in} and R_{in-cir} be the input value of R_{in}. So, we update R_{in} as:

For iter < 3 and $abs(\frac{R_{in_{ext}} - R_{in_{exp}}}{R_{in_{exp}}}) > 0.01$:

$$R_{in_{cir}} = R_{in_{cir}} * \frac{R_{in_{exp}}}{R_{in_{ext}}} \quad and \quad iter++$$

Following the same convention,

$g_{m_{exp}} = 1.2 * gain_{exp}/R_l$

For iter < 5 and $op1db_{ext} < op1db_{exp}$:

$$W_{cir} = W_{cir} * (\frac{g_{m_{exp}}}{g_{m_{ext}}})^2 \quad and \quad iter++$$

5.2 Main-Optimization Stage

Here, we implement gradient descent algorithm to find the optimal set of circuit parameters. We have already seen how to define the loss functions here and how to implement multivariate gradient descent. Here, we outline the equations based on which the circuit parameters are updated in every iteration.

Loss L = $Loss_{gain} + Loss_{op1db} + Loss_{am\_pm\_dev} + Loss_{I_o} + Loss_{p\_higher\_harms}$

We need to implement:

$$\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} L(\theta_1, \theta_2, ...\theta_n)$$

The individual loss weights are chosen in such a way that the net loss function is dimensionless. So, we normalise the gradient calculation to match units and modify the equation to get:

$$\theta_i = \theta_i(1 - \alpha \frac{\partial L(\theta_1, \theta_2, ...\theta_n)}{\partial \theta_i / \theta_i})$$

In gradient calculations, $\frac{\partial L}{\partial \theta}$ is approximated as $\frac{\Delta L}{\Delta \theta}$ with $\Delta\theta = 0.001\theta$. Here, 0.001 is the Delta_threshold. The algorithm allows us to change this value if we need to calculate more sensitive gradients.

We start the pre-optimization stage with a set of input parameters that satisfy all the output constraints. So, the main aim of this optimization loop is to reduce the power consumption while retaining all the output specs.

We make the design choice that adhering to the output specs is more important than reducing power consumption. So, in a particular iteration, if the output specs are not met, we update the loss function to be independent of the power consumption and focus only on meeting the specs again. It tries to reduce the bias current only when the output specs are met. But the algorithm also has flags to modify this design choice and focus on all the loss contributing parameters at once.

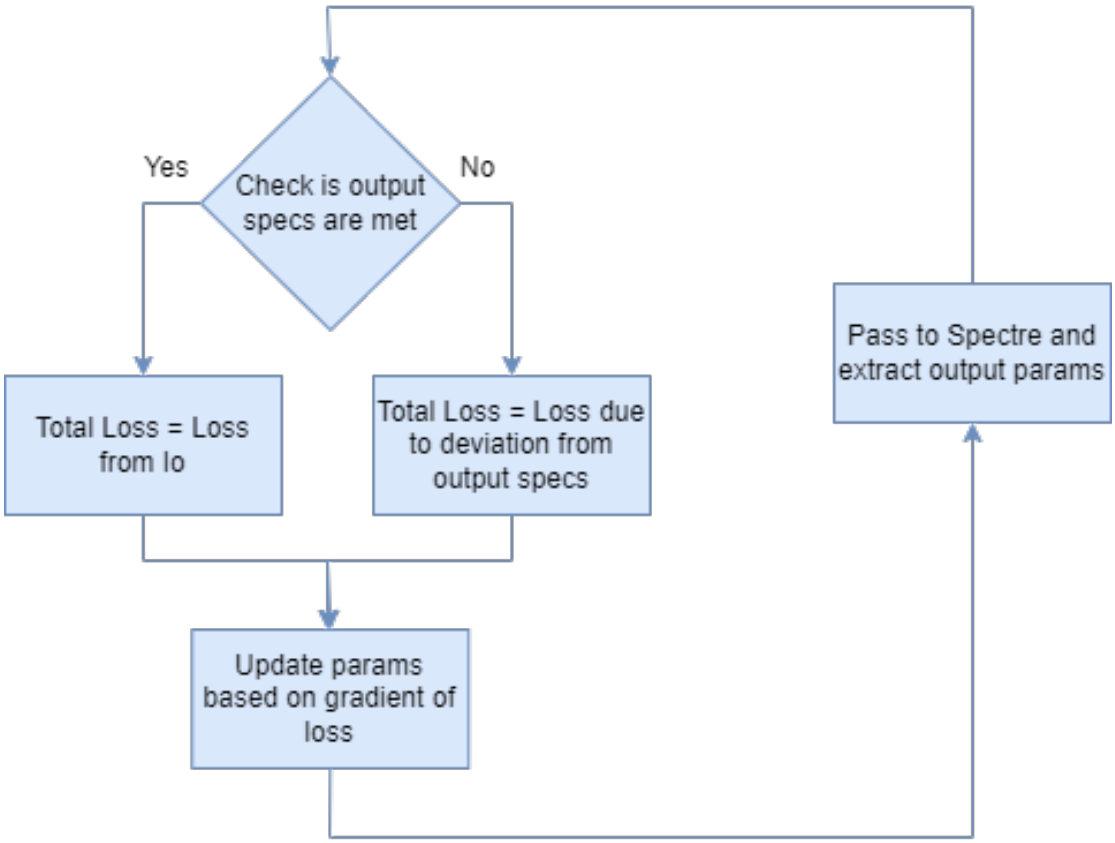

Figure 5.2: Flowchart showing loss type definition

5.3 Choosing best operating point

The gradient descent algorithm will tries to minimise the net loss and there could exist a case where there is a trade-off between one of the output specs and power drawn. So, the lowest loss might occur in an iteration where all the output specs are not met. To avoid such scenarios, we first filter out those iterations where loss due to output parameters is zero and power drawn is the only contributor to loss. In those iterations, we choose the one with the lowest power drawn as the best operating point for the circuit.

CHAPTER 6

ANALYSIS RESULTS & DISCUSSION

6.1 Single Point Analysis of Power Amplifier

The output specs of concern to us are the ss gain, the op1dB, ip1dB, AM-PM deviation and the power consumption. We obtain each of these output parameters by separate analysis done on the PA.

6.1.1 DC Analysis

We use this to set the DC operating point for the circuit. We save all the node voltages, drain current drawn, gm, gds, parasitic capacitances, vth, vdsat and the operating region. We print all these outputs into a "dc.out" file and use basic file I/O to read the values from the file.

6.1.2 AC Analysis

We use the ss AC analysis for 3 purposes:

- To find i/p and o/p voltages and hence the ss gain (mag and phase)
- To find the effective input AC resistance, $R_{in_{ext}}$
- To find the effective load AC resistance, $R_{l_{ext}}$

6.1.3 HB Analysis

We use the large signal HB analysis for:

- Analysing the gain and output power at different levels of input power given to the circuit
- Finding where the AC gain falls by 1dB, use that to measure the corresponding ip1dB and op1dB

- Analysing the power delivered at different frequencies at the ip1dB i/p level to estimate non-linearity at the output signal
- Measuring the drain current drawn at ip1dB power level to estimate the max power drawn by the PA from the supply in its range of operation
- Measuring the phase of the o/p voltage signal for different levels of i/p power to calculate the AM-PM deviation spec of the PA

To achieve this purposes, we need to run harmonic balance analysis at different input power levels multiple times. This can be achieved through a direct "hbsweep" analysis or a running "hb" analysis inside a parametric sweep in spectre or by externally varying the input power and just running basic "hb" analysis in spectre. We address all these 3 scenarios in upcoming sections.

6.1.4 XDB Analysis

We also use the inbuilt "xdb" gain compression analysis to calculate the ip1db and op1db values. We observe that the xdb analysis converges faster and gives more accurate results than the hb sweep analysis as it intelligently varies the step sizes with large initial steps and reduces the step size as it senses the AC gain to reduce. This allows it to have a very small step size near the ip1dB point and hence gives accurate outputs.

But we can use this analysis only on the ideal circuit and not on the circuit with real passive components. So, we only run xdb analysis for a way to verify the values obtained from manual hb sweep.

All the initial analysis is done for a single-ended PA. The table of expected output specs is mentioned here again:

Table 6.1: Single-Ended PA Output Constraints

Operating frequency	2GHz
SS Voltage Gain (gdB)	$\geq$ 6dB
AM-PM Deviation	$\leq$ 5 deg
O/P 1dB Compression Point (op1dB)	$\geq$ 10dBm

6.2 Tweaking of Hand Calculations

Based on the hand calculations detailed here, the component parameters obtained are tabulated below. We observe that they don't match the gain specs chosen because the g_m is only 0.5-0.6x the expected value. So we choose to increase the bias current and width keeping the I/W ratio constant such that the node voltages remain constant and g_m alone is increased.

Table 6.2: Chosen circuit parameters

Parameter	After Hand Calc	After DC oppoint analysis
Rin	50	50
Rb	5000	5000
Rl	12.5	12.5
Ld	10nH	10nH
C1	3.2pF	3.2pF
C2	1.25nF	1.25nF
W	105um	210um
Io	20mA	40mA

But this approach increases the bias current drawn and hence the net power consumption increases too. So instead of increasing Io, we increase W even further to achieve the required g_m.

Table 6.3: Chosen circuit parameters after tweaked Hand Calc

Parameter	After tweaked Hand Calc
Rin	50
Rb	5000
Rl	12.5
Ld	10nH
C1	3.2pF
C2	1.25nF
W	420um
Io	20mA

6.3 Pre-Optimization Results

The updated hand-calculations are added in the code. After Hand-calculations, 5 iterations are done in the pre-optimization stage to increase MOSFET width until the g_m is 1.2x the expected value until the output 1dB power specification is matched.

Note: Io is the input bias current to the current mirror. Drain current is the DC current drawn by the amplifying MOSFET.

Table 6.4: Chosen circuit parameters after Pre-Optimization

Parameter	After tweaked Hand Calc	After Preoptimization
Rin	50	50
Rb	5000	5000
Rl	12.5	12.4
Ld	10nH	10nH
C1	3.2pF	3.18pF
C2	1.25nF	1.27nF
W	420um	416um
Io	20mA	20mA

Table 6.5: Output Specs after Pre-optimization

Parameter	Pre-optimization output
SS Voltage Gain	5.47dB
AM-PM Deviation	0.034 degrees
O/P 1dB Compression Point	10dBm
Drain Current	26.4mA

6.4 Optimization Results

With the above chosen parameters after pre-optimization, we implement gradient descent algorithm.

6.4.1 Optimizing Ideal Single-Ended Circuit

Version 1

Table 6.6: Obtained Circuit Parameters after Optimization V1

Parameter	Value
Rin	50
Rb	5020
Rl	46.07
Ld	11nH
C1	6.97pF
C2	2.8nF
W	1.27mm
Io	1.15mA

Table 6.7: Output parameters after Optimization V1

Parameter	Optimization output
SS Voltage Gain	6.57dB
AM-PM Deviation	0.14 degrees
O/P 1dB Compression Point	10.02dBm
Drain Current	3.6mA

Observations:

- There is almost a 100% error in the current drawn by the current mirror MOSFET and the amplifying MOSFET.
- The configuration of the class-A PA limits efficiency to 50%. So, supplying a 10dBm output power demands atleast 20mA supply current. This implies that the power drawn from V_{dd} varies a lot when we sweep the input power over a large range.
- While the hand calculations assumed all the MOSFETs to be in saturation region of operation, the optimization pushes them into Sub-threshold region to consume less power.

- External bounds must be imposed on certain circuit parameters during the optimization loop: $R_l < 50\Omega$ and $L_d < 10nH$

Version 2

Corrections made:

- Report Id current from DC oppoint analysis and from avg current in 0th harmonic in HB analysis as the drain current will be different for a ss input and a larger ac input.
- Implement it by calculating the current drawn from supply at 1dB compression point and use it in grad descent. In saturation region, the hb analysis gave relatively larger DC supply current than DC analysis. But in chosen convergence point after optimization, both analysis give similar current values indicating improved linearity.

Table 6.8: Obtained Circuit Parameters after Optimization V2

Parameter	**Value**
Rin	50
Rb	5410
Rl	38.8
Ld	5.56nH
C1	89.7pF
C2	35.9nF
W	1.48mm
Io	18.1mA

Table 6.9: Output parameters V2

Parameter	**Optimization output**
SS Voltage Gain	16dB
AM-PM Deviation	0.019 degrees
O/P 1dB Compression Point	10.0dBm
I/P 1dB Compression Point	-6dBm
Drain Current DC	31.8mA
Drain Current HB	26.9mA

Observations:

- Imposing a constraint on R_l changes the convergence point of the optimization.
- Currently Ids is around 27mA with only 20mA needed for a 10dBm output power. So, we try reducing the bias current alone and see how the output parameters are affected.
- We try for 9dBm output power instead of 10dBm and see if it helps in reducing current.

Version 3

Modifications:

- Set Rbias as max(5k, 50*gate impedance). Gate impedance comes from the gate capacitor with cap/width around 1.2fF/um.
- Make the coupling capacitors as a function of R and not as an optimising parameter.
- The gain is a strong function of the input coupling capacitor and currently a threshold of 2000 is used for it. But a threshold of 100 seems fine for output coupling.

Table 6.10: Obtained Circuit Parameters V3

Parameter	Value
Rin	50
Rb	2450
Rl	46.8
Ld	8.23nH
C1	64.7pF
C2	0.169nF
W	1mm
Io	17.6mA

Observations:

- Setting o/p 1dB constraint as 9dBm doesn't let the op1db value go beyond 9.2 in most iterations since it is the controlling spec in the loss function. So, optimising for a 10dBm o/p 1dB constraint and then choosing the best iteration with op1dBm above 9dBm gives better results.
- In the previous case, Ids of 26.9mA gave a op1db of 10dBm giving an efficiency of 37.17%, consuming 35% more Ids than required.

Table 6.11: Output parameters V3

Parameter	Optimization output
SS Voltage Gain	15.9dB
AM-PM Deviation	0.054 degrees
O/P 1dB Compression Point	9.15dBm
I/P 1dB Compression Point	-6dBm
Drain Current DC	28.4mA
Drain Current HB	21.7mA

- With reduced op1dB, Ids of 21.7mA gives op1dB of 9.15dBm again giving an efficiency of 37.9% consuming 32% more current.
- Sweeping Io doesn't seem to significantly affect any of the critical specs at the output. But for lower Io values, the non-linearity of the device increases.
- While analysing sensitivity, having low MOSFET widths coupled with a low bias current matches the expected specifications with high efficiency but the device is highly non-linear. There is also on a trade-off on AM-PM deviation but the values are still well within the specified limit. But to retain linearity, we impose constraints on the power dispersed in higher harmonics at the output. This helps the optimization converge towards a high ss gain operating point while the supply power drawn varies minimally with a large variation in input power.

Version 4

Incorporating all the above modifications into the algorithm, the finally chosen circuit parameters after optimization for a single-ended Power Amplifier are tabulated below.

Table 6.12: Obtained Circuit Parameters V4

Parameter	Value
Rin	50
Rb	2359
Rl	34.95
Ld	5.99nH
W	1.38mm
Io	17.8mA

Table 6.13: Output parameters V4

Parameter	Optimization output
SS Voltage Gain	15.4dB
AM-PM Deviation	0.018 degrees
O/P 1dB Compression Point	10dBm
I/P 1dB Compression Point	-6dBm
Drain Current DC	31mA
Drain Current HB	31.27mA

6.4.2 Optimizing with TSMC Resistors & Capacitors

We retain the same circuit parameters obtained from optimizing the ideal circuit in version 4 and observe the output parameters when we replace all the ideal resistors and capacitors used in the circuit with their corresponding TSMC models.

Table 6.14: Output parameters with TSMC RC

Parameter	Optimization output
SS Voltage Gain	16.23dB
AM-PM Deviation	0.02 degrees
O/P 1dB Compression Point	10.73dBm
I/P 1dB Compression Point	-6dBm
Drain Current DC	35.44mA
Drain Current HB	34.55mA
gm value	503mS

When we run the complete optimization loop on this circuit using real R, C blocks, the circuit converges towards a non-linear operating point. This result prompted us to include the additional term in the loss function penalising the distribution of output power in higher harmonics.

Table 6.15: Obtained Circuit Parameters with TSMC RC Optimized

Parameter	Value
Rin	50
Rb	42100
Rl	35.6
Ld	2.26nH
W	395u
Io	2.61mA

Table 6.16: Output parameters with TSMC RC Optimized

Parameter	Optimization output
SS Voltage Gain	6.44dB
AM-PM Deviation	2.77 degrees
O/P 1dB Compression Point	10dBm
I/P 1dB Compression Point	2.5dBm
Drain Current DC	5.46mA
Drain Current HB	15.2mA
Rl observed	47.1 ohm

6.4.3 Optimizing with TSMC Inductors

Similarly, we replace the ideal inductor connected to the drain of the MOSFET with a spiral inductor from TSMC. In this scenario, we observe an interesting error in the TSMC inductor PCell. We had originally used the inbuilt "hbsweep" to run hb analysis on the circuit multiple times with different input powers. And this sweep analysis is necessary to find certain output parameters like op1db and AM-PM deviation. The "hbsweep" analysis works as expected on the ideal circuit and when using TSMC blocks for resistors and capacitors. But when we include the TSMC inductor block, the analysis hangs mid-way. We work-around this error by externally varying the input power and avoiding any inbuilt "sweep" analysis functions in Spectre and this solution gives us the expected results but as a trade-off, consumes significantly more time than using "hbsweep" because a large portion of the runtime is to call spectre and analyse its outputs as compared to running the actual analysis. We also use the general purpose models (gp) from the TSMC65 PDK for all our analysis. Another observation is that this issue in hbsweep only occurs in the gp models and not on the low power (lp)

models. So, another possible solution is optimizing for many iterations on the lp models, get a close operating point and run a few more optimizing iterations around the obtained point on the gp models. This idea is still not yet validated using experiments and is one of the future objectives of the project.

Simulating with the same set of circuit parameters obtained from v4 Optimization, we get:

Table 6.17: Output parameters with TSMC RLC

Parameter	Optimization output
SS Voltage Gain	15.90dB
AM-PM Deviation	0.388 degrees
O/P 1dB Compression Point	9.47dBm
I/P 1dB Compression Point	-6dBm
Drain Current DC	36.15mA
Drain Current HB	38.99mA
gm value	511mS

Optimizing the complete real circuit, we again converge at a non-linear operating point.

Table 6.18: Obtained Circuit Parameters with TSMC RLC Optimized

Parameter	Value
Rin	50
Rb	43200
Rl	34.7
Ld	21.1nH
W	676u
Io	7.38mA

Table 6.19: Output parameters with TSMC RLC Optimized

Parameter	Optimization output
SS Voltage Gain	6.54dB
AM-PM Deviation	4.62 degrees
O/P 1dB Compression Point	10dBm
I/P 1dB Compression Point	2.5dBm
Drain Current DC	13.6mA
Drain Current HB	29.1mA
gm	173mS

6.4.4 Optimizing Differential Circuit with TSMC Components

Here, we combine 2 single-ended PA circuits to achieve a pseudo-differential configuration of a PA. The updated output specifications for the differential PA are re-tabulated here.

Table 6.20: Differential PA Output Constraints

Operating frequency	2GHz
SS Voltage Gain (gdB)	$\geq$ 12dB
AM-PM Deviation	$\leq$ 5 deg
O/P 1dB Compression Point (op1dB)	$\geq$ 13dBm

Manually modifying the circuit parameters we obtained in the optimization v4, we get:

Table 6.21: Circuit Parameters - Differential PA

Parameter	Value
Rin	50
Rb	2359
Rl	69.9
Ld	11.98nH
W	1.38mm
Io	17.8mA

Table 6.22: Output parameters

Parameter	Ideal	TSMC RLC
SS Voltage Gain	22.25dB	21.55dB
AM-PM Deviation	0.37 degrees	1.895 degrees
O/P 1dB Compression Point	13.77dBm	14.199dBm
I/P 1dB Compression Point	-6dBm	-5dBm
Drain Current DC	35.44mA	36.15mA
Drain Current HB	34.56mA	37.324mA
gm value	503mS	511mS

Using Centre-Tapped Inductor

Then we replace the individual inductors connected to the drains of the MOSFETs on either side with a centre-tapped inductor. Then, we expect the signals on either side to couple with each other and the effect of the parasitics to be enhanced. We also impose a 10nH upper limit on the centre-tapped inductance value. Hence we get:

Table 6.23: Output parameters with Centre-tapped inductor

Parameter	Optimization output
SS Voltage Gain	21.86dB
AM-PM Deviation	1.569 degrees
O/P 1dB Compression Point	14.09dBm
I/P 1dB Compression Point	-5dBm
Drain Current DC	33.128mA
Drain Current HB	31.52mA
gm value	479.51mS

Using a Differential Matching Network

In the above chosen configuration, the PA requires to drive a 70Ω output impedance. Since the input impedance of the next stage is 100Ω, we connect a high-pass MN at the load side as mentioned in this schematic. We calculate the MN parameters to be:

Using this MN, the obtained output parameters are:

Table 6.24: MN Parameters - Differential PA

Parameter	Value
Lsrc	2.274nH
Lload	2.6526nH
Cmn	2.92pF

Table 6.25: Output parameters with MN

Parameter	70ohm Rl	100ohm Rl + MN
SS Voltage Gain	22.25dB	21.01dB
AM-PM Deviation	0.373 degrees	1.279 degrees
O/P 1dB Compression Point	13.77 dBm	12.83 dBm
I/P 1dB Compression Point	-6 dBm	-6.5 dBm
Drain Current DC	35.44 mA	35.44 mA
Drain Current HB	34.56 mA	42.46 mA
gm value	503.9 mS	503.9 mS

Here, these output parameters are obtained for the circuit with ideal components. We observe that circuit using the MN consumes more power when delivering op1dB than the one without a MN.

Then we replace all the passive components with their real TSMC equivalents and add $L_{src}, L_{load}, C_{mn}$ to the set of optimizing parameters. The optimization converges to the following circuit parameters:

Table 6.26: Circuit Parameters - Differential PA with TSMC RLC Optimized

Parameter	Value
Rin	50Ω
Rb	5850Ω
Rl	100Ω
Ld	3.46nH
W	461μm
Io	16.7mA
Lsrc	2.6nH
Lload	2.01nH
Cmn	1.87pF

Table 6.27: Output parameters with TSMC RLC Optimized

Parameter	Optimization output
SS Voltage Gain Magnitude	18.93dB
SS Voltage Gain Phase	-126°
AM-PM Deviation	3.26degrees
O/P 1dB Compression Point	13.85dBm
I/P 1dB Compression Point	-3dBm
Drain Current DC	24.2mA
Drain Current HB	35.22mA
gm	227mS
Zl	80.3Ω

Here we observe that while separating the PA without the MN, we converged at a smaller value of R_l. But while we optimise with the MN, we converge at a slightly larger value with a significant complex impedance. A consequence of this is that we get a phase difference between the input and output signals. We try penalising this phase difference in the loss function but we couldn't achieve convergence for a max. allowed phase difference of 10 degree. This occurs because the MN has a purely resistive impedance only for a constrains set of values for $L_{src}, L_{load}, C_{mn}$ and independently optimizing the 3 parameters make it tougher to fix on those values that give a resistive Z_{in}. But as long as the phase difference caused is consistent across the operating power range and frequency band, the functionality isn't affected and we proceed with it.

6.5 Output Trends

By running single-point analysis on the chosen set of circuit parameters, we plot the significant trends observed in output parameters w.r.t input power which is swept over a fixed range.

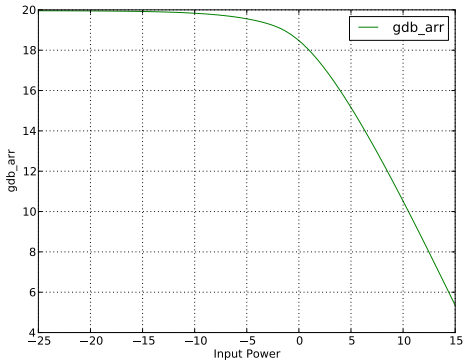

Figure 6.1: gain dB: Output Trend

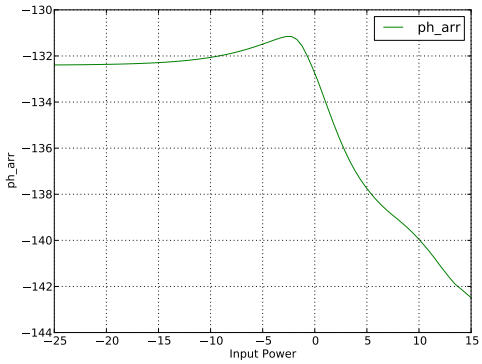

Figure 6.2: gain phase: Output Trend

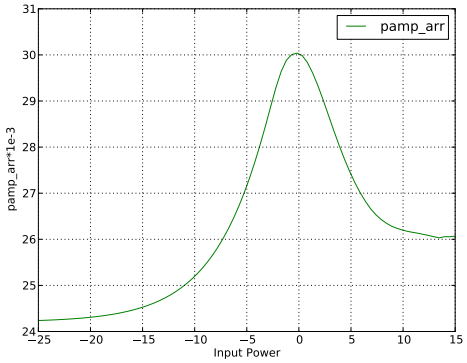

Figure 6.3: Power consumed by single-side PA: Output Trend

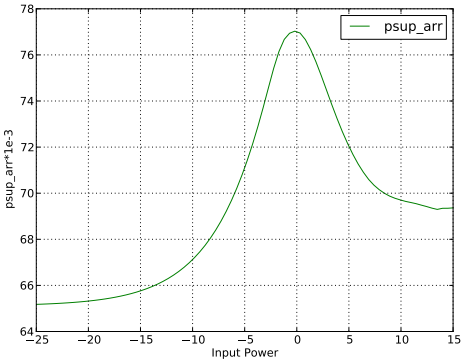

Figure 6.4: Power consumed by complete PA: Output Trend

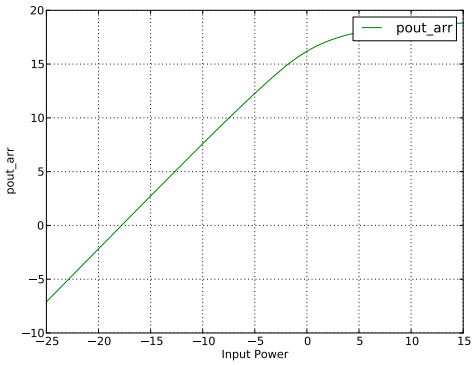

Figure 6.5: Pout: Output Trend

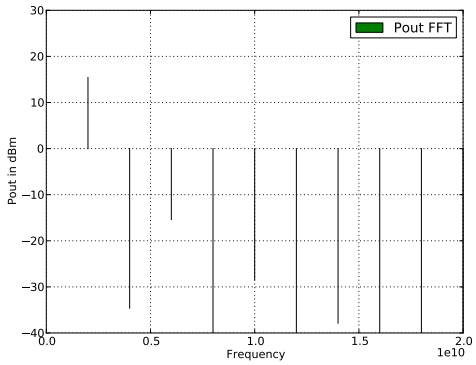

Figure 6.6: Power Spectra: Output Trend

6.6 Temperature Analysis Results

We perform temperature sensitivity analysis on the circuit for the chosen set of component parameters. We vary the temperature between -40°C and -120°C and also vary the bias current to analyse whether we can vary the current as a function of temperature to retain the performance.

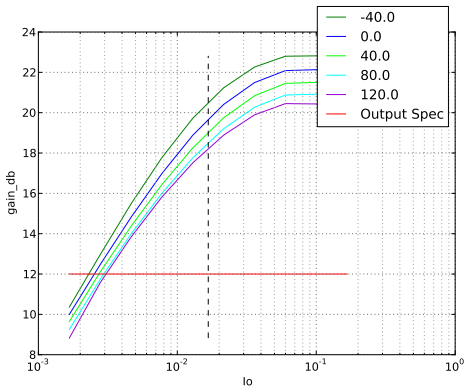

Figure 6.7: gain dB: Temp Analysis

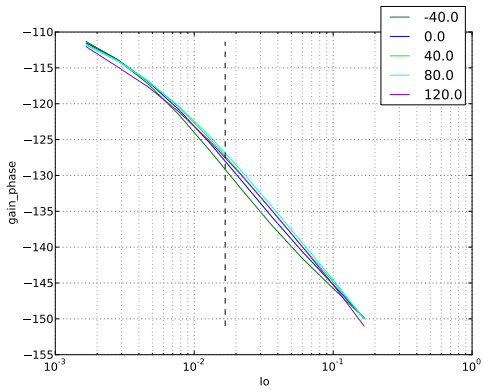

Figure 6.8: gain phase: Temp Analysis

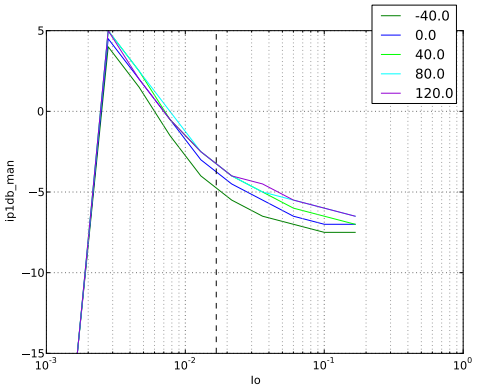

Figure 6.9: ip1dB: Temp Analysis

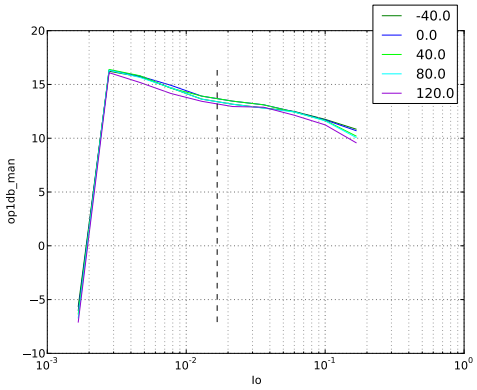

Figure 6.10: op1dB: Temp Analysis

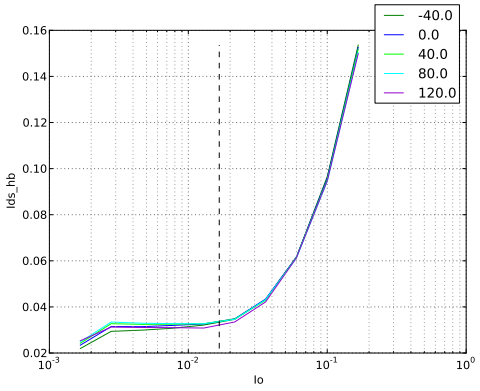

Figure 6.11: Ids at ip1dB: Temp Analysis

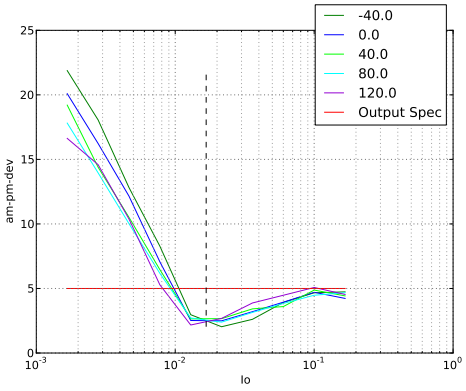

Figure 6.12: AM-PM-Dev: Temp Analysis

As expected, all the output parameters are well within the constraints over the operating range of temperature at the fixed bias current. So, there is no necessity to vary the bias current to retain performance.

6.7 Process Analysis Results

We also perform process corner analysis on the circuit. We analyse the performance over even (tt, ff, ss) and odd (sf, fs) corners. We also sweep the temperature to see the combined variation in the circuit.

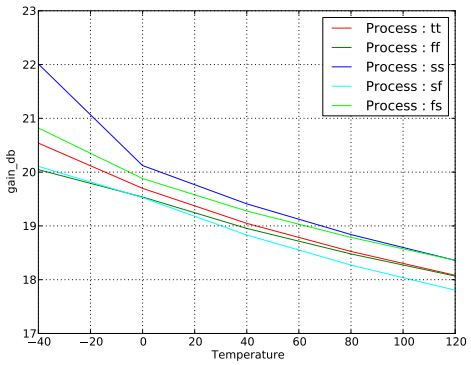

Figure 6.13: gain dB: Proc Analysis

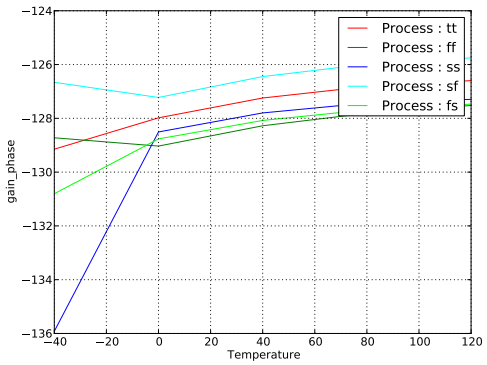

Figure 6.14: gain phase: Proc Analysis

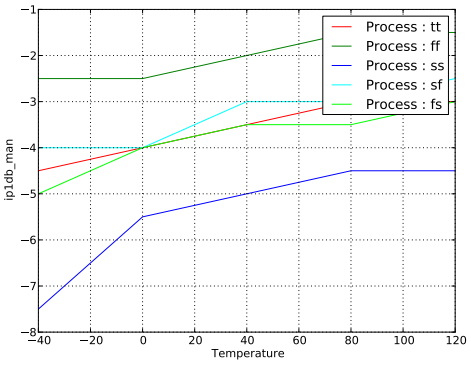

Figure 6.15: ip1dB: Proc Analysis

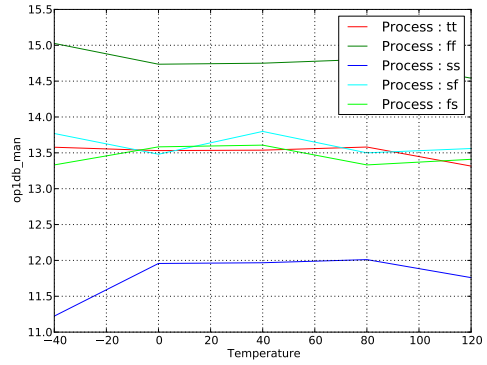

Figure 6.16: op1dB: Proc Analysis

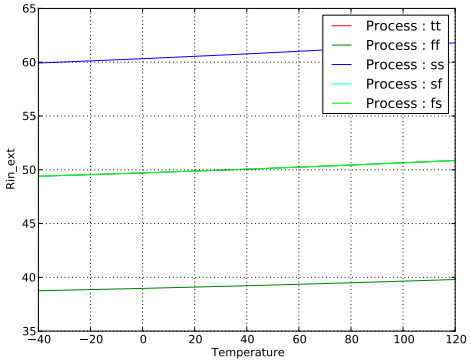

Figure 6.17: Rin: Proc Analysis

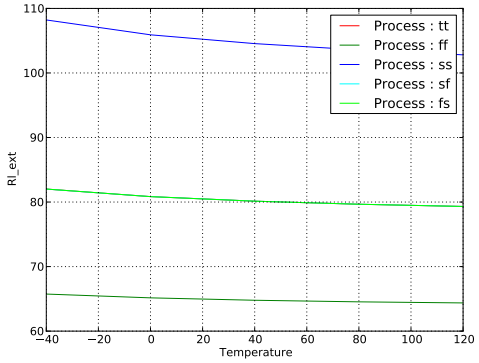

Figure 6.18: Rl: Proc Analysis

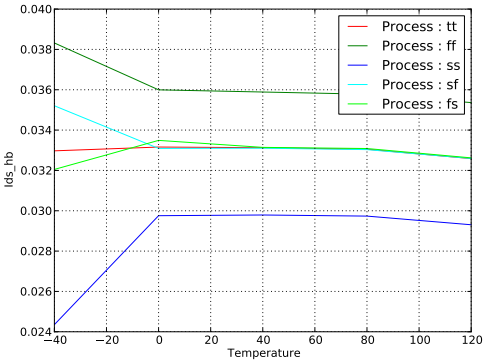

Figure 6.19: Ids at ip1dB: Proc Analysis

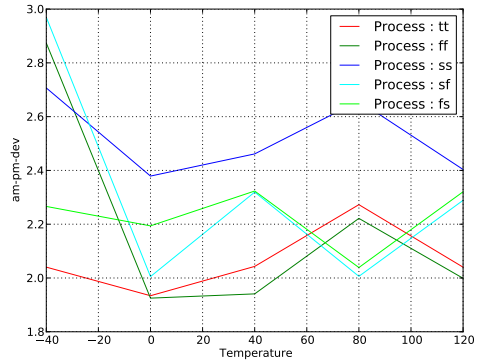

Figure 6.20: AM-PM-Dev: Proc Analysis

The current algorithm calculates the loss function by taking the worst output parameter among the 3 frequencies of interest and comparing it with the output specifications. The next stage in the process is trying to optimize the circuit not just w.r.t the bandwidth but also w.r.t process and temperature variations. We would need to simulate the circuit at all corners and choose the worst parameter among those for loss function calculation. We are currently working on parallellising the code to include this functionality as sequentially running this code for all possible process-temp combinations for every iteration would blow up the running time.

CHAPTER 7

CODE DOCUMENTATION

7.1 Project Directory Tree

```
Automatic RFIC Synth
├── spectre_single_point.py
├── ckt_optimizer.py
├── PA
│   ├── hand_calc.py
│   ├── pre_optimization.py
│   └── spectre.py
├── Netlists Ref
│   ├── circ.scs
│   ├── Single_ended
│   └── Differential
├── Ckt Analysis
│   ├── process_analysis.py
│   ├── temp_analysis.py
│   ├── sensitivity_analysis.py
│   └── circuit_parameter_analysis.py
├── Ckt Optimization
│   ├── optimization.py
│   ├── grad_descent.py
│   └── loss_func.py
├── MOS Files
│   ├── tsmc_components.py
│   └── TSMC65.txt
└── Spectre Run
    └── Tx
        └── spectre_run.tcsh
```

The optimization algorithm runs on a python + spectre back-end. Python codes are used for interfacing with the user, with the spectre simulator and for processing all the obtained data. There are two main codes/analysis in the project.

The spectre_single_point.py code is used to simulate a PA circuit for a single specific set of circuit parameters obtained from the user. It invokes updates the input circuit parameters in the netlist, invokes spectre, performs all the mentioned analyses on the circuit and extracts the output parameters and prints them back to the user.

The ckt_optimizer.py code triggers the optimization algorithm. It takes as input the output constraints and the optimization parameters, updates netlists, invokes spectre, extracts output parameters, calculates the loss function and its gradient and updates the circuit parameters iteratively until it achieves a set of optimal parameters.

7.1.1 PA

This folder contains all the files specific to the PA circuit.

hand_calc.py

This is invoked in the automatic initial point and initial point update stage. Functions in this file are called by the pre_optimization function.

pre_optimization.py

This file takes the decision between using manual initial point from the user or to calculate automatic initial point and invoke hand_calc function

spectre.py

This file is the backbone of the algorithm. This defines a PA circuit class and all functions are defined within this class. It contains functions to arrange the experiment directory, update circuit parameters to netlist, modify the tcsh files to invoke spectre, extract parameters from spectre log files and process them to get output parameters. This invokes 3 spectre calls parallelly to simulate the circuit at f_o, $f_o - \Delta f$ and

$f_o + \Delta f$. For this, we organize the running directory by creating 3 separate folders with individual netlist copies, one for each frequency so that they don't interfere with each other. We also modify the code with multithreading so that the same function can be called multiple times parallelly without a hit on running time.

7.1.2 Netlists Ref

This contains the original netlist files for the circuit. The current version of the netlist to be used is stored as "circ.scs". The spectre.py code copies this file into 3 different directories (one for each frequency) and updates the values of circuit components defined as parameters in the netlist.

Older versions of the netlist files used are stored in the folders "Differential" and "Single_Ended" for reference.

7.1.3 Ckt Analysis

This directory contains codes used to automate different kinds of analysis like:

- circuit parameter analysis: A chosen parameter is swept over a specified range and trends in other output parameters are measured
- temperature analysis: The temperature of the circuit is swept and output parameters are analysed
- process analysis: The output parameters are analysed over tt, ss, ff, sf and fs process corners
- sensitivity analysis: A chosen parameter is varied by a small amaount and the extent by which the output parameters change are measured

7.1.4 Ckt Optimization

This directory contains all the codes to optimize the circuit by implementing gradient descent algorithm.

optimization,py

This file contains invokes the optimization algorithm. It first calls the pre-optimization function and then invokes the gradient descent algorithm. It also has the data logging code to store all the parameters during the optimization loop.

grad_descent.py

This file contains code to invoke the loss function calculator, calculate its gradient and find the extent by which the circuit parameters are to be updated. It also has the code to update the learning rate, loss type and other optimization parameters and plot all optimization trends at the end.

loss_func.py

This file contains the loss function calculator and the code to find the best possible solution among the qualifying iterations.

7.1.5 MOS Files

This contains the path files to all the TSMC65 model libraries stored in the central system. It also has the codes to choose the parameters for TSMC R,L,C components.

tsmc_components.py

This file contains the code to calculate parameters for:

- Resistors: Sheet width and length for a given resistance value
- Capacitors: MIMCAP width and length for a given capacitance value
- Inductor: The entire customised inductor finder algorithm is implemented here

7.1.6 Spectre Run

As explained in the previous section, we run 3 parallel simulations for the same circuit at different frequencies. To invoke spectre CLI, we need to run a tcsh shell and set the

environment variables. To automate this and run it parallelly, we define 3 different tcsh files to copy the netlist into the right directory and invoke spectre to run the analysis.

7.2 Circuit Parameter Storage and Access

A majority of the code in this project is to access data and process it. It could be from the log files from spectre analysis or the output parameters in every iteration and trends formulated based on these parameters. We use two different ways to store data during the optimization loop. The data stored during runtime is stored in dictionaries. Once the optimization loop is over, we store the data from all the iterations in CSV files.

7.2.1 Data stored in dictionaries

We use nested dictionaries to store all the circuit related and optimization related information. The main dictionaries in use are:

- Circuit Initialization Parameters: An input dictionary that stores values like MOS model parameters, simulation directory paths, netlist parameters (sweep limits, number of harmonics etc.), circuit temperature, process corner etc.
- Optimization Input Parameters: An input dictionary that stores values regarding optimization like loss weights, learning rate, iterations, optimizing parameters, initial point selection and simulation and netlist parameters.
- Circuit Parameters: A dictionary defined inside the circuit class. It stores the values of all circuit components. The values stored in this dictionary map to corresponding parameters in the netlist through the write_dict function.
- Extracted Parameters: A dictionary defined inside the circuit class and stores all the extracted output parameters at all 3 frequencies of analysis.

7.2.2 Data stored in CSV files

After the optimization loop is over, the following CSV files are written:

- alpha parameters: This stores the learning rate of every optimizing parameter in every iteration
- circuit parameters: This stores every circuit component parameter in every iteration

- loss: This stores the loss function value and its individual components in every iteration.
- loss slope: This stores the gradient of loss w.r.t every circuit parameter in every iteration
- output parameters: This stores the output parameters in every iteration

Figure 7.1: Sequential diagram showing the hierarchy of functions defined

CHAPTER 8

CONCLUSION

We have implemented Gradient Descent algorithm to optimize the differential Power Amplifier Circuit successfully. We also include a Matching Network and analyse if its better to optimize them individually or together. This can be scaled to the TX network. This PA will have a Gilbert Cell Mixer driving its input and its output will be connected to the Antennae. So, we need to scale the algorithm to optimize the complete TX side of the wireless system given a set of output constraints for the communication network.

We currently fix the topology of the PA and optimize only for the chosen set of circuit components in that topology. We need to scale the algorithm on that front to choose the optimal topology. We can start by choosing a set of 2-3 probable topologies, optimize them individually and then choose the best topology among them based on power consumed or any other chosen constraint.

We also run process and temperature analysis on the obtained optimal circuit and observe that the output performance dips below the constrained limits on select process-temperature corner combinations. We can choose to include these corner cases in the optimization algorithm itself. We will initially run the original optimization algorithm for 150-200 iterations and converge near the optimal point. Then, we can modify it to include P-T corner cases and do a complete optimization for the next 20-30 iterations, We suggest such an approach because completely optimizing the circuit over all corners for 200 iterations would take a significantly large amount of time and memory.

All of this analysis is performed on the schematic of the PA. But as mentioned in the introduction, the performance of RF circuits is closely coupled with the effect of layout parasitics. So, we need to check for cases where the optimal circuit parameters chosen for the schematic might have larger layout parasitics and its performance will be lower than another set of circuit parameters with poor schematic performance but has a more compact layout.

These are the future scope of this project.