A Wide Tuning-Range mm-Wave LC-VCO Sized Using Evolutionary Algorithms

A Thesis

Presented in Partial Fulfillment of the Requirements for the Degree
Honors Research Distinction in the The Ohio State University

By

Matthew R. Belz,

Undergraduate Program in Electrical and Computer Engineering

The Ohio State University

2019

Research Distinction Committee:

Dr. Waleed Khalil, Advisor

Dr. Steven Bibyk

Dr. Tawfiq Musah

Abstract

Designing a LC Voltage Controlled Oscillator (LC-VCO) for mm-Wave frequencies requires a careful balance of interdependent design parameters. The losses due to passive elements dictate the required cross coupled pair transconductance ($\boldsymbol{g_m}$), which in turn affects the tuning range via fixed capacitance. As such, the design process requires significant engineering time. An optimization methodology using a genetic algorithm is proposed to optimize component selection for use in the LC-VCO. The design for the LC-VCO is broken into pseudo-independent sub-modules to allow the designer greater control and to allow the optimization to benefit from manual circuit intuition. Performance of the components chosen by the genetic algorithm is verified using a circuit simulator to achieve a center frequency of 29 GHz with a 15.8 GHz tuning range. The simulated phase noise performance is -103.2 dBc/Hz using a 10 MHz frequency offset.

Acknowledgments

First I would thank my parents who have always supported my scientific endeavors and academic pursuits. They always pushed me to do my best work and without them I wouldn't have made it this far.

All the members of the Circuit Lab for Advanced Sensors and Systems (CLASS) have not only been incredible mentors, but also great friends. Our casual conversations have given me many useful insights. I also have learned so much from the now graduated students Dr. Matt LaRue, Dr. Luke Duncan, and Dr. Jamin McCue who all were very open to my questions when I was just starting in the group. Dr. Shahriar Rashid helped me understand some of the basics of RF design during our chats and whiteboard sessions about VCOs. I really appreciate your willingness to help me understand the concepts at a fundamental level.

I would especially like to thank Gus Fragasse for his help on my VCO modeling paper. There were a few times when the simulations weren't working, and I thought I would not make the deadline. Gus patiently helped me debug, and encouraged me to keep plugging away—for this I am very grateful.

When I first started making my own electronic gadgets as an electrical engineering major Dr. Steve Bibyk helped guide me when I got stuck. Thanks for all the support over the years.

I also would like to thank Dr. Tawfiq Musah for serving on my defense committee.

Finally, I would like to thank my advisor Dr. Waleed Khalil. As an undergrad at such a large institution it is easy to get lost and Dr. Khalil was always there to help me make decisions about class choices, graduate school, and how to develop my skills to be successful in my later career. Dr. Khalil is passionate about electronics and also cares deeply about each of his students, there's not much more one can wish for in an advisor.

Table of Contents

List of Tables

List of Figures

Figure **Page**

Chapter 1: Introduction

As demand increases for mm-Wave communication links such as the emerging 5th Generation cellular standard (5G) and automotive RADAR, CMOS-based wide tuning range (TR) mm-wave Voltage Controlled Oscillators (VCOs) are positioned to become an integral part of next generation System on Chips (SoCs) [1, 2]. The increasing use of wireless technologies such as Wi-Fi, Bluetooth, and Cellular along with the demand for higher data rates has led to spectrum crowding. Moving up in frequency to the mm-Wave bands is a current trend since it will allow for higher data rates due to the larger availability of open spectrum. The frequency range for mm-Wave is roughly defined between 30 and 300 GHz.

Designing an integrated voltage controlled oscillator (VCO), a critical component to any wireless system, requires many engineering hours to tune so it will meet the required specifications. Like many analog circuits, the design process for the VCO necessitates a careful balance of interdependent factors. If one component is changed such as the varactor to increase the tuning range, several other parameters will need to be updated such as the transconductance from the cross coupled pair as a larger varactor will reduce the parallel resistance of the tank. This will in turn increase the tank capacitance and reduce both the operating frequency and tuning range [1]. This is one example of how the design of a VCO becomes a circular relationship and can be difficult to break.

While there are many different types of voltage controlled oscillators, the topology used in this work uses an LC tank and a negative resistance element. The LC-VCO approach is widely used for high frequency applications and is well suited for mm-Wave operation [1]. This work is focused on a design automation approach to reduce the number of design iterations and overall design time for the LC-VCO.

1.1 Clock Generation in Hardware

An accurate frequency reference is required for data transmission and reception in both wireline and wireless domains. Crystal oscillators are used to generate frequency references with accuracy on the order of 0.1-10's of parts per million. Nevertheless, crystal oscillators resonate at a particular fixed frequency, well below the single GHz range. In order to generate an adjustable frequency reference suitable for high-frequency data transmission, a phase locked loop (PLL) is often used. The PLL seen in Figure 1.1, takes an ultra-stable reference, often a precision crystal, and then uses a feedback loop to reduce phase errors from the VCO. The difference between the RF output phase and the reference is measured using the error detection block in Figure 1.1. The phase difference is measured by the phase detector (PD) and the PLL corrects for this difference using the charge pump (CP). The CP generates a control voltage which will adjust the frequency of the VCO. The output frequency from the VCO is divided for the error detection stage using the divide by N block where N can be either integer or fractional. After an initial startup period, the PLL is ideally locked to the accuracy of the reference signal and produces an output waveform

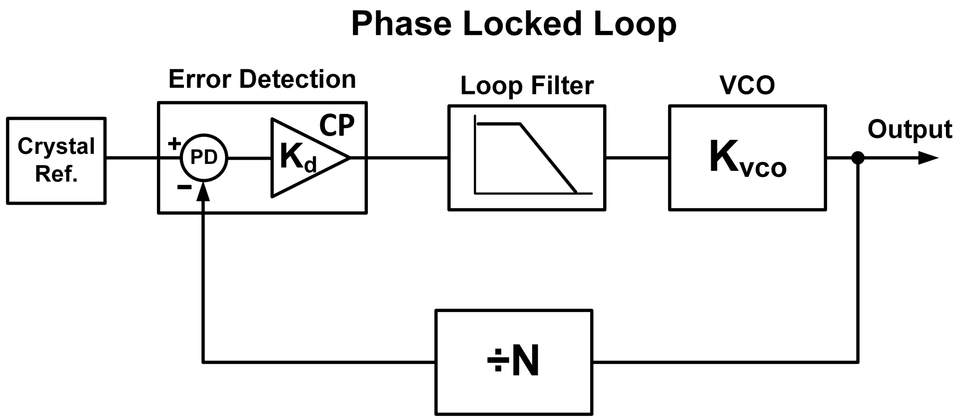


Figure 1.1: Simplified PLL Model

suitable for high-speed clocks and data transmissions. The PLL can then be used as a local oscillator (LO) for wireless transmitters such as the homodyne transmitter seen in Figure 1.2. The homodyne transmitter takes a baseband signal and directly upconverts to the RF frequency for use in wireless standards such as Bluetooth.

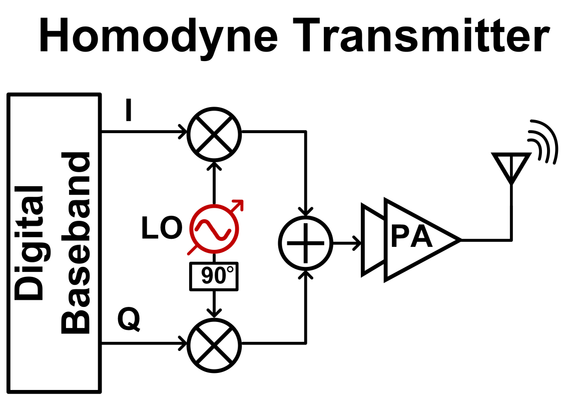


Figure 1.2: Homodyne Transmitter Model

The gain of the VCO is denoted K_{vco} and represents the slope of the line when the output frequency is plotted vs. the control voltage. In other words, K_{vco} shows how sensitive the output frequency is for a given change in the control voltage. Since the oscillation frequency is $\frac{1}{2\pi\sqrt{LC}}$ either the inductor or capacitor can adjust the operation frequency. In practice, due to the difficulty of adjusting on-chip inductors, an adjustable capacitor is used for the frequency adjustment. In this work an accumulation mode varactor is used to change the capacitance of the tank and thus the operation frequency [3].

Chapter 2: LC-VCO With C-DAC Optimization

The work in [4] presents a multi-objective algorithm to optimize the trade-off between power consumption and phase noise reflected in the Figure of Merit (FOM) for the VCO. After optimization, the FOM was comparable with previous state of the art designs. However, the VCO tuning range is a critical parameter of performance and its optimization has not been explored.

In this work, an optimization technique is presented which includes a segmented capacitive digital to analog converter (C-DAC). In this topology, the C-DAC is used to achieve low phase noise and wide tuning range simultaneously [5]. The framework selected for optimization is a multi-objective genetic algorithm (GA), which is used to optimize the TR and tank quality factor by determining the best component sizes for the LC-VCO topology as presented in Figure 2.1. Unlike the TR parameter, the PN of the VCO is not automatically optimized using a cost function, but rather it is optimized by using a small value of inductance and using a C-DAC to ensure a low K_{vco}. The addition of PN into the optimization is discussed in the future work section.

The optimization methodology proposed in this work has a runtime on the order of one minute compared to several hours reported previously [4, 6]. The runtime

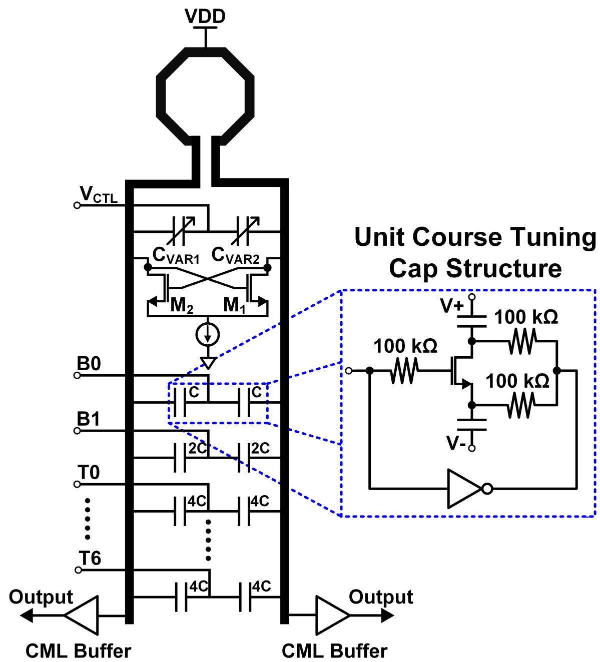


Figure 2.1: LC-VCO Architectusing a 10 MHz frequency offset.ure

is reduced since the evolutionary algorithm optimizes based on cost surfaces developed using prior simulation data rather than using the simulator to evaluate the cost function for each generation. This allows for an optimization based on simulation data for each block rather than relying on analytical equations that suffer from gross simplifications.

2.1 LC-VCO Design Methodology

A bottom-biased LC-VCO topology, presented in Figure 2.1, is the subject of the optimization in this work. A symmetric, center-tapped inductor is used for the inductive element in the VCO. At mm-Wave frequencies, only a single turn is required, thus, eliminating this variable from the search space. Additionally, since the PN is inversely proportional to the inductance of the tank inductor, the value of inductance is minimized within the geometric allowances of the process design kit in a pre-processing optimization step [1].

The use of a single varactor element to provide a tuning range sets a limit on the achievable PN performance due to amplitude modulation to phase modulation (AM to PM) conversion from the large tuning curve slope K_{vco} [7]. Reducing K_{vco} while maintaining a large tuning range is achieved by digitally switching fixed capacitors for coarse tuning, and using a varactor for fine tuning. To ensure operation across process voltage and temperature (PVT) as well as parasitic variations, the varactor is sized for 30 to 50 percent frequency overlap between the tuning curves.

The number of tuning curves are determined prior to running the evolutionary algorithm and represents a trade-off between tuning resolution and tuning range. While increasing the number of tuning curves results in a finer coarse tuning resolution, it also adds parasitic capacitance and inductance due to the feed line network [5]. This parasitic capacitance becomes significant at mm-Wave frequencies and ultimately limits the tuning range. Additionally, the use of more tuning curves will increase the frequency tuning time since the search space to find the correct tuning curve is increased. The optimum number of tuning bits is typically between four and six.

In this work, we use a 5-bit C-DAC resulting in 31 coarse tuning curves. The C-DAC is implemented as a segmented DAC where the two least significant bits (B0-B1) are binary to save area while the three most significant bits (T0-T6) are thermometer coded to reduce mismatch between tuning curves [8]. The schematic for a digitally switched capacitor cell used in the C-DAC is shown in the inset of Figure 2.1. A trade-off exists between the switch transistor size and the unit capacitor quality factor.

When the transistor is very wide, the resistance is low, however, the gate capacitance also increases lowering the ratio between C_{ON} and C_{OFF} states leading to a smaller tuning range. To balance this trade-off, we desire the C_{ON}/C_{OFF} ratio to be around three.

The cross coupled NMOS pair is designed to compensate for the losses due to the passive elements of the tank in the form of a negative resistance. To satisfy the Barkhausen criterion for oscillation across PVT variations, a design margin of two is used. In short channel technologies, there is only a weak dependence on current to increase the g_m, therefore the width of the cross coupled pair devices is used to obtain the necessary transconductance [1]. The resulting capacitance from the cross-coupled pair (C_{xcpl}) is added as fixed capacitance in the GA evaluation. This fixed capacitance accounts for shifts in center frequency and tuning range. This is a critical step since C_{xcpl} accounts for around 38% of the total capacitance of the tank at 30 GHz [1], [9].

2.2 Genetic Algorithm Design

Genetic algorithms are a class of evolutionary algorithms that optimize a population of individuals stochastically without the use of hill-climbing methods such as gradient descent or Newton's Method. In each generation a population is created, in this work 200 each generation consists of 200 individuals. This type of optimization algorithm is well suited to problems such as circuit design, which is characterized by a large search space, especially when a topology is not specified, with many local minima. The goal of a genetic algorithm is to find the region of highest population fitness and therefore allows for suboptimal individuals to exist in order to further explore the search space and avoid becoming stuck in local minima. After discovering the region of lowest cost, or equivalently the highest fitness, a gradient-descent method can be used to find the lowest cost individual in that generation. Despite the fact that genetic algorithms can not provide a provably optimal solution, they are well suited to circuit design problems due to the complexity and size of the solution space [6]. Individual parameters are encoded into a chromosome which can be a binary string or floating point value. Due to the possibility of Hamming cliffs, floating point and integer values are used for chromosome encoding in our algorithm design [10]. Selection is performed using a stochastic uniform distribution with a selection of parent chromosomes modified using crossover and mutation. Crossover allows for the combination of current generation genes to form new children. Using crossover, a chromosome with a suboptimal gene (e.g. varactor width) could be paired with a different complementary gene (e.g. varactor number of fingers) to generate a superior overall solution. Crossover does not add new individual genes to the pool but just switches existing genes from different chromosomes to form a new individual. Mutation introduces small changes in the chromosome to explore the search space further. These mechanisms are shown visually in Figure 2.3. For the genetic algorithms used in this study, the most fit individual is carried over to the next generation unchanged (elitism). This ensures that the best fitness of the population is always increasing even if the population overall has decreasing fitness as local minima are encountered.

The cost function is the core of any genetic algorithm and determines how individuals are selected and propagated through new generations. In the genetic algorithm design, the cost function is minimized or equivalently the fitness is maximized. A weighted sum of fitness functions are used to create a multi-objective optimization

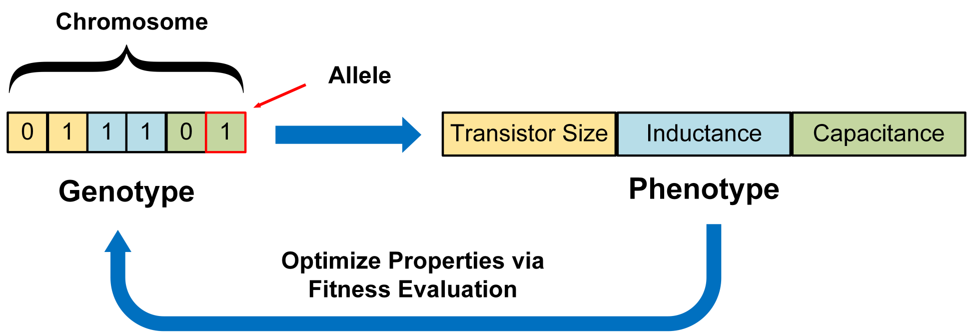


Figure 2.2: Genetic Algorithm Encoding Process

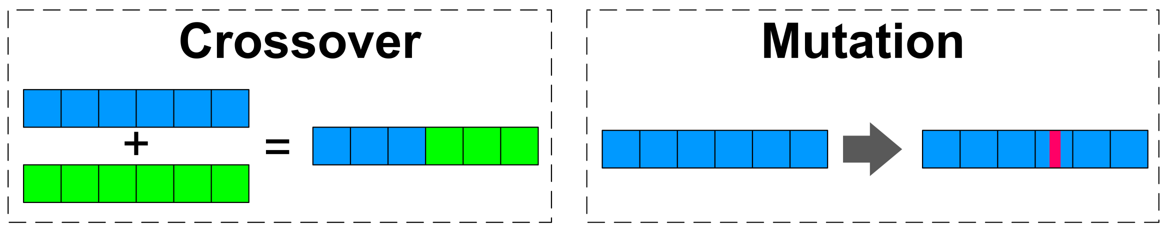


Figure 2.3: Crossover and Mutation Illustration

algorithm [11]. The fitness functions used for this optimization approach are shown in the genetic algorithm application section.

2.3 VCO GA Optimization Setup

To optimize the VCO, a genetic algorithm is used to solve the multi-objective optimization problem by setting fitness goals for each sub-block (C-DAC, cross coupled pair, inductor, and varactor) of the VCO according to the design practices discussed in section 2.1. A population size of 200 individuals was used and the individual with the highest fitness is carried into the next generation. The bounds for the optimizaion are set according to the allowed geometry range from the process design kit (PDK) or a known feasibility region whichever is more restrictive. Parameters such as the number of fingers for the varactor are restricted to integers, while continuous valued variables (e.g. varactor width) are optimized as floating point numbers.

2.3.1 Cost Surface Generation

To construct cost surfaces for the genetic algorithm, we utilized simulation data from a 45nm CMOS SOI process to create fit surfaces as a function of geometric design parameters. While predictive process design kits and extensions of long channel models can be applied to understand trends in short channel devices, the viability of a component selection algorithm requires simulation with empirically-backed models. Many component selection methods utilize circuit simulators with in-loop optimization to meet this requirement [4,6]. The drawback is that in-loop simulation methods require significant overhead since electromagnetic and circuit simulations are much more computationally intensive compared to equation based evaluation. Additionally, this computational load can not be performed in parallel since the next circuit configuration chosen by the GA is based on the results of the previous iteration. To address this problem, we have collected a large multidimensional matrix of simulation test data that is then interpolated using linear or spline methods to produce a continuous surface as a function of the geometric dimensions. Simulation times for the sweeps used to generate the cost surfaces are all under 5 minutes for each sub-block and the GA optimization runs in under one minute.

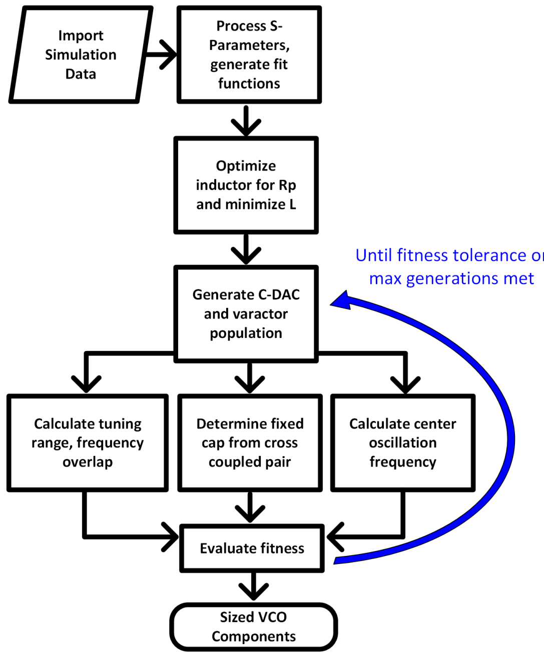
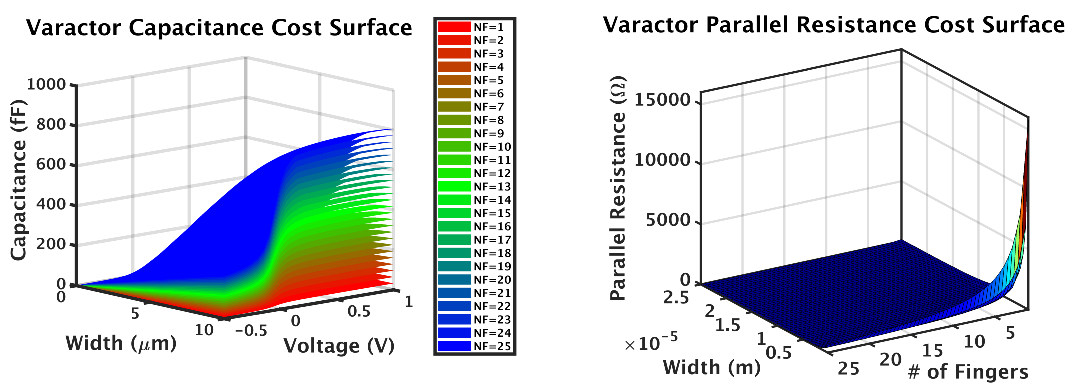


Figure 2.4: VCO Optimization Algorithm



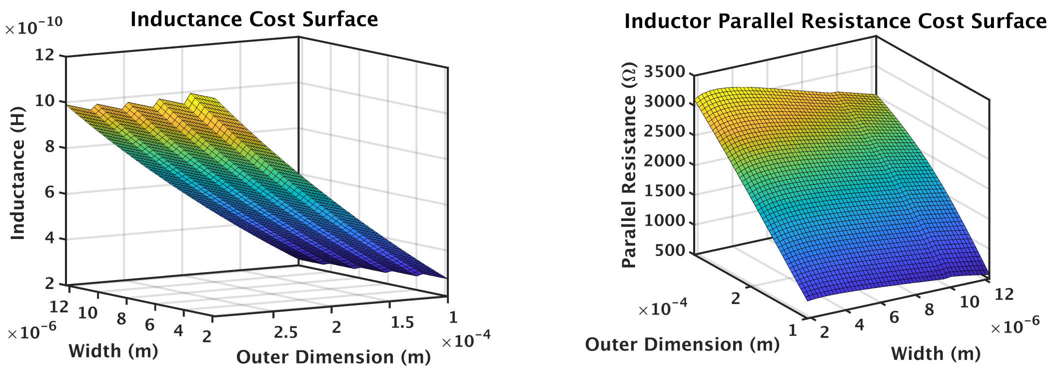
(a) Varactor CV Characteristic

(b) Varactor R_p Function

Figure 2.5: Varactor Cost Surfaces

2.3.2 Cost Function Derivation

A flowchart representation of the algorithm used to optimize the VCO is shown in Figure 2.4. A linear combination of cost functions are used to enable multi-objective



(a) Inductance Cost Surface

(b) Inductor R_p Function

Figure 2.6: Inductance Cost Surfaces

optimization that align with key objectives from the manual design [11]. This approach is used to tune the VCO more precisely than could be achieved by tuning a FOM as the sole cost function. Since the desired oscillation frequency is known prior to design, the data extracted from the circuit simulator is taken at this frequency to reduce the dimensionality of the data. However, some parameters such as $R_{p,L}$ are averaged over the expected tuning range to account for changes during tuning.

The genetic algorithm attempts to minimize the cost function through crossover and mutation of the parameters in the vector $\boldsymbol{X}$, which is shown in (2.1). The inductor value and $R_{p,L}$ are determined in a pre-processing step using the cost surfaces for the inductor.

$$\boldsymbol{X} = [NF_{sw}, L_{cap}, W_{var}, NF_{var}] \tag{2.1}$$

NF_{sw} is the number of fingers for the digital switching transistor inside the metal-oxide-metal (MOM) capacitor structure and L_{cap} is the length of the MOM capacitors.

Increasing the width of the MOM capacitor significantly degrades the R_p, so it is fixed throughout the optimization. W_{var} is the varactor width and NF_{var} is the number of fingers for the varactor.

For cost functions, binary true or false values are not particularly useful for a genetic algorithm as they do not indicate how close the current solution is to the ideal case. To give the algorithm feedback during the iteration process, the absolute value of the difference between the current solution and target value is used. This strategy was adopted for the overlap between curves, center oscillation frequency and C_{on}/C_{off} ratio. As the GA improves the solutions, the absolute difference between the current and target solutions will approach zero, therefore minimizing the cost function.

To determine the overlap between coarse tuning curves, the top two frequency curves are used since they represent the worst case overlap. To calculate the parallel resistance of the tank, the worst case is taken when all digital caps are ON. To compute the required transconductance from the cross coupled pair, the parallel $R_{p,ON}$ combination is used, which also reveals the resulting transistor width and parasitic C_{xcpl}.

The cost function that is used to optimize the VCO is shown in Equation 2.2. The α values are used to normalize the cost function such that large values like tuning range do not dominate the cost. If the cost function is not properly normalized only a select few of the optimization objectives will be met. For example since TR is on the order of 1E9, smaller optimization targets such as the C_{on}/C_{off} ratio will be ignored.

$$Cost = \alpha_1 |3 - \frac{C_{on}}{C_{off}}| + \frac{\alpha_2}{R_{p,cap}} + \frac{\alpha_3}{R_{p,var}} + \alpha_4 |28GHz - f_{mid}| + \alpha_5 |0.4 - overlap| + \frac{\alpha_6}{TR} \quad (2.2)$$

The tuning range is specified from the middle region (50% varactor) of the top and bottom curves. The use of the middle instead of the extremes of the tuning curves was chosen as to minimize the impact of K_{vco} since the tuning range could be extended by making K_{vco} very large when measuring at the extreme points of the curves.

Once the resonant tank is determined, the cross-coupled pair can be sized based on the total parallel resistance of the tank. In this case the design margin in Equation 2.3 is two. The required g_m is used along with the fit functions of Figures 2.7 and 2.8 to find the width of the cross coupled pair devices and the added parasitic capacitance respectively. This parasitic capacitance is voltage dependent so this parasitic analysis could be made more accurate by running a transient and then averaging the parasitic capacitance, for this study the C_{xcpl} was determined using an ac analysis.

$$g_m = \frac{2}{R_{p,T}} \tag{2.3}$$

2.4 Results

The operation of the LC-VCO optimized and sized using the GA has been tested in a circuit simulator to verify the performance given as output from the GA. The sizes chosen for the VCO core by the GA are shown in Table 2.1. To illustrate the tuning characteristics for the GA design VCO, the tuning curves are presented in Figure 2.10. The simulated TR aligns with the range given from the GA simulation as they are both 15.8 GHz. It is important to note that the tuning range will be reduced once extracted layout parasitics, which are outside the scope of this work, are added. The tuning range will still be maximized through this algorithm, but the

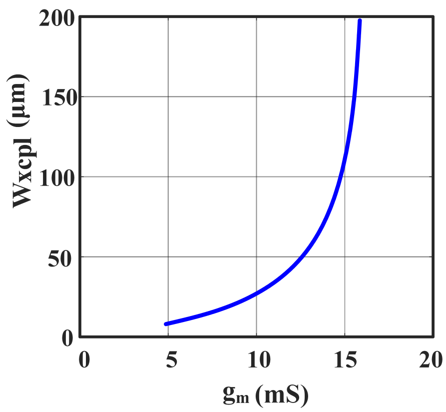


Figure 2.7: Fit Surface for Cross-Coupled Pair Width

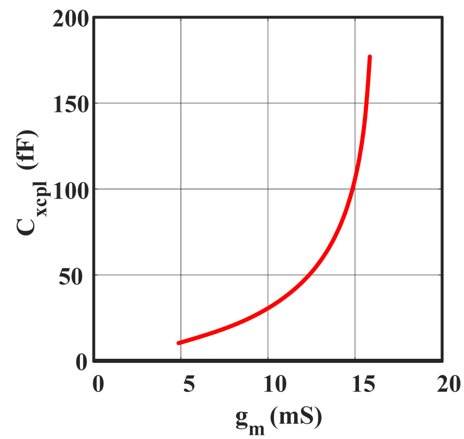


Figure 2.8: Fit Function for Cross Coupled Pair Parasitic Capacitance

inclusion of extracted layout parasitics will provide for a more accurate value for the tuning range. The PN is presented in Figure 2.11. Since the PN is not explicitly optimized in the cost functions of the GA, it is expected that increased performance can be achieved if this objective is included in a future version. At an offset of 10 MHz the PN performance is -103.2 dBc/Hz. Using the equation for $\mathrm{FOM_T}$ shown in Equation 2.4, the LC-VCO achieves a $\mathrm{FOM_T}$ of -181.2 dBc/Hz when simulating PN at 10 MHz offset and a center frequency of 28 GHz.

$$FOM_T = PN - 20log\left(\frac{f_0}{\Delta f}\cdot\frac{TR}{10}\right) + 10log\left(\frac{P}{1mW}\right) \tag{2.4}$$

The generation of interpolated fitness functions, particularly in the varactor, is attributed to the slight error between the frequency predicted in the GA model vs. the simulated LC-VCO, which was approximately 0.6 GHz for the maximum frequency. Since the GA needs to fit functions across a high-dimensional matrix there is some

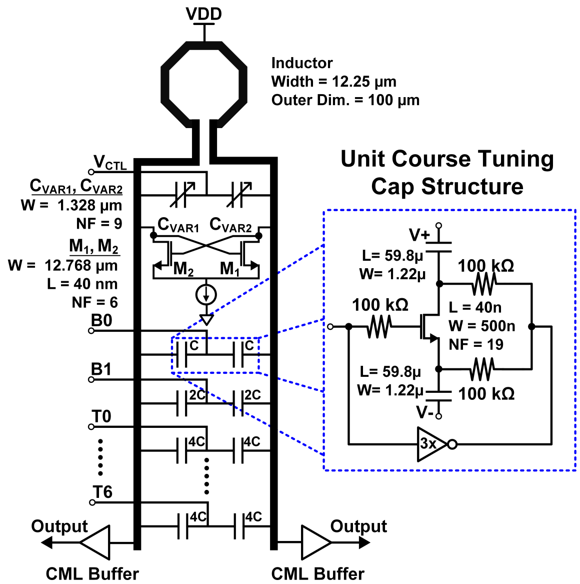


Figure 2.9: Sized VCO Design

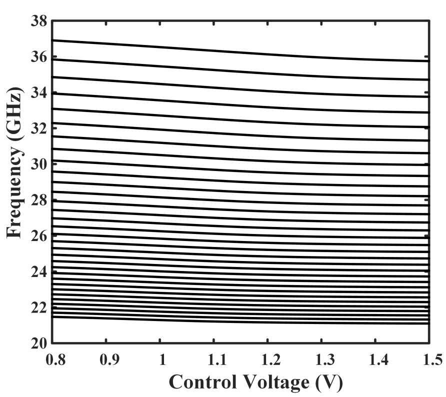


Figure 2.10: Simulated Tuning Curves of the GA-optimized VCO

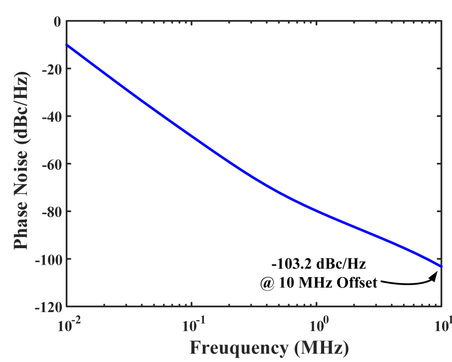


Figure 2.11: Simulated VCO Phase Noise Performance

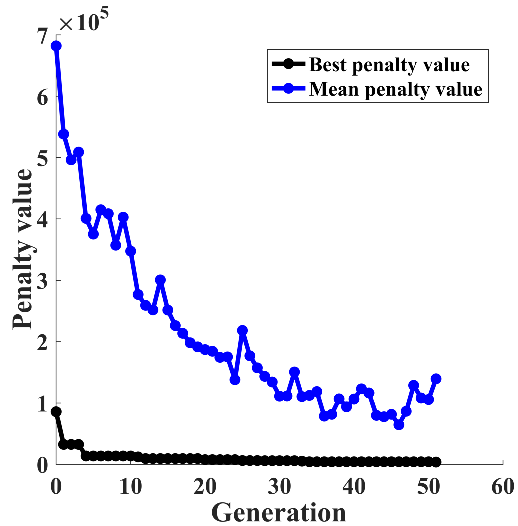


Figure 2.12: Genetic Algorithm Convergence

error in any individual curve, and is observed to be limited to a difference of under 5 fF. The algorithm is found to converge to an optimal region rather quickly as evidenced by the convergence plot presented in Figure 2.12. The lowest penalty (cost) is monotonically decreasing since the individual with the highest fitness is always carried into the next generation. The simulated startup and steady state oscillation of the VCO designed by the GA is presented in Figure 2.13 and shows the positive and negative output nodes.

2.5 Conclusion

A genetic algorithm using a weighted sum of interpolated cost surfaces is shown to produce a sized LC-VCO designed for applications requiring a wide TR. This LC-VCO uses a C-DAC for coarse tuning to allow for a wide TR and low PN. Cost functions

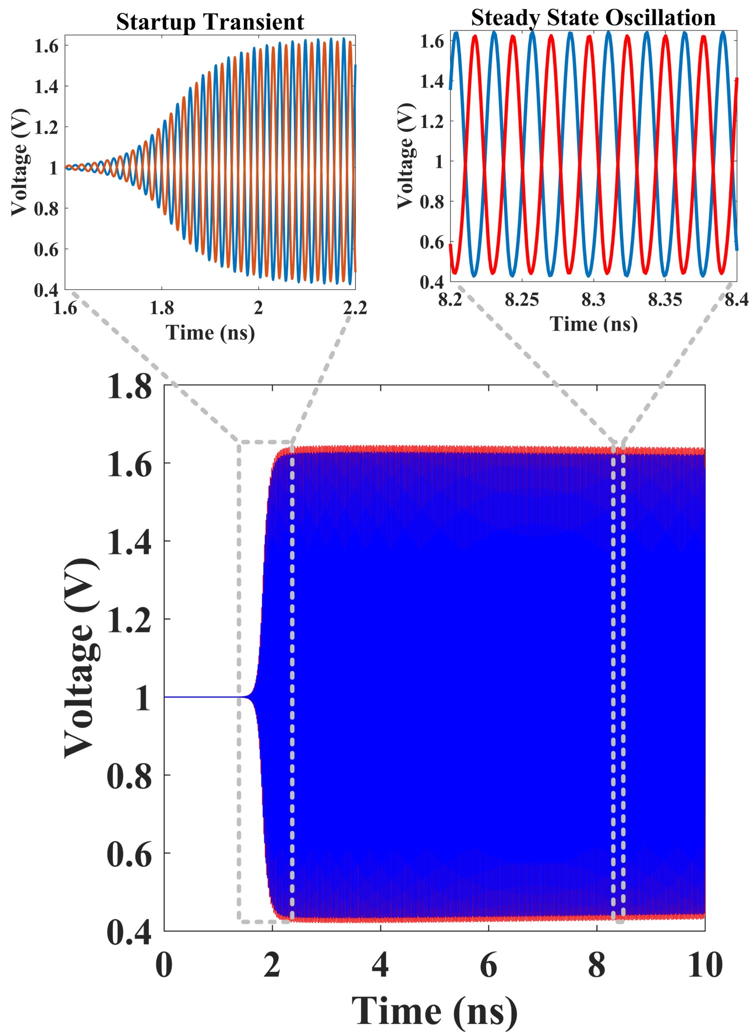


Figure 2.13: Oscillation of VCO

Table 2.1: Optimized VCO Component Sizes

Variable	**Optimized Result**
NF_{sw}	19
L_{cap}	59.8μm
W_{var}	1.328μm
NF_{var}	9
W_{xcpl}	12.77μm
Ind. Outer Dimension	100μm
Ind. Width	12.25μm
Inductance	130.7 pH
Unit Switch Cap (C)	12.4 fF
Varactor Low	14.79 fF
Varactor High	46.6 fF

based on simulation data are used to provide accurate modeling for component selection while allowing for fast fitness evaluation. Techniques from manual design are translated into a genetic algorithm to reduce the search space and aid in convergence. The resulting sized VCO is then simulated in a circuit simulator to verify the results obtained through the genetic algorithm. This simulation shows the TR is found to be 15.8 GHz and the PN at 10 MHz offset is -103.2 dBc/Hz. The PN is not modeled in MATLAB however the TR model in MATALB matches almost exactly with the simulated tuning range. Enhancements to the optimization algorithm could improve accuracy by including fitness metrics for phase noise and additional parasitics due to layout.

Table 2.2: Fitness Objectives and Results

Variable	**Goal**	**Simulation Result**
C_{on}/C_{off} ratio	3	3.84
Tuning Range	Maximize	15.8 GHz
Center Frequency	28 GHz	29 GHz
$R_{p,var}$	Maximize	467.98 Ω
$R_{p,cap}$	Maximize	5.34 kΩ
Current Consumption	Minimize	5 mA

Table 2.3: Performance Comparison to Manually Designed LC-VCOs

Ref.	Tech.	Center Freq. (GHz)	PN @ 10 MHz Offset (dBc/Hz)	TR (%)	Output Power (mW)	$\mathrm{FOM_T}$
[12]	32 nm SOI	24.7	-127.3	22.9	24	-188.6
[13]	32 nm SOI	28	-110	22	31	181.2
[14]	45 nm CMOS	24.8	-121	25	12.1	-186.0
[15]	130 nm CMOS	40	-115	27	12	-184.9
[16]	32 nm CMOS	39.9	-118	31.6	9.8	-190.1
This Work	**45 nm SOI**	**28**	**-103.2**	**56.4**	**5**	**-181.2**

Chapter 3: Conclusions and Future Work

3.1 Summary

This work has presented a methodology for rapidly designing a VCO suitable for use at mm-Wave frequencies. The segmented C-DAC topology described in Chapter 3 allows for a wide TR while maintaining low PN. Cost surfaces based on simulation data allow for rapid fitness evaluation while still maintaining a high level of accuracy. Additionally, this allows for rapid re-designs in new technology nodes by updating the cost surface data from the circuit simulator.

3.2 Broader Impacts

The design of a VCO is a time consuming process and also requires expert knowledge to complete a working design. In industry, time-to-market is often a critical factor and using an evolutionary algorithm such as presented in this work can reduce the design time by determining a good starting point for further manual tuning if required. This can especially be helpful when a new technology node is used since the designer will need to learn new guidelines on how the device sizes correlate to performance.

Defense contractors and government labs also may find this work useful since their focus is based on missions which are constantly changing. The ability to design a custom ASIC without expert designers in every RF sub-block is a topic of current research through the DARPA electronics resurgence initiative (ERI). In defense applications, some performance degradation from the state-of-the-art may be tolerable as long as the mission requirements are met.

Both industry and government labs could benefit from the ability to use artificial intelligence for circuit design by reducing the required engineering hours and free up designers to consider more creative architectures instead of manually perturbing the component sizes to determine the optimal conditions.

3.3 Future Work

As technology nodes continue to scale, simulations based on schematic designs are becoming increasingly poor at predicting performance as layout parasitics become more dominant. Currently the GA uses parametric sweep data from a schematic-level simulation and assumes proper layout will give the designer similar (but degraded) performance. If the layout is generated along with the schematic representation, the parasitics due to the layout can also be considered leading to a more accurate indication of performance once the device is fabricated. Additionally, for mm-Wave frequencies the inductor and feed line network should be measured using an electromagnetic (EM) simulator and included during the optimization. For this work it was assumed that the PDK inductor simulation was sufficiently accurate and the feed line parasitics were negligible. The Berkeley Analog Generator would be a good platform to build a layout aware GA since it has interfaces to both circuit and EM simulators using layout extracted parasitics in simulation runs [17].

While the GA optimizes for the component sizing, there are still several aspects which must be designed manually such as the biasing circuitry. The bias circuits not only have a significant impact on the noise of the device [18], but they also impact the figure of merit through the power consumption. If the bias current is added as a parameter in the algorithm, the power consumption could also be optimized automatically instead of tuning based on designer intuition. By optimizing the bias current the PN can also be minimized by operating at the boundary between the current limited and voltage limited operating regions for the VCO. Additionally second order effects such as supply pushing are not considered, but are a critical specification for many applications.

Appendix A: Collaboration

This appendix discusses the collaboration during the completion of this work. The circuit field is moving away from absolute maximum circuit performance to generator based design where a circuit generator can design several different custom circuits based on an initial topology to serve different application requirements. My work has been some of the first at the CLASS group to address this growing need in the circuit community for rapid design and optimization. Dr. Shahriar Rashid helped me to understand the details of traditional circuit design, and in turn, I was able to further his understanding of how the design space can be mapped into an optimization framework. A similar collaboration occurred with Saeed Alzahrani who also specializes in VCO design. Over the summer I also did some preliminary work for VCO optimization with layout parasitics using the Berkeley Analog Generator. This required installing and customizing environments on our servers to run. Dr. Luke Duncan later used the BAG environment that I helped setup for his own projects.

Appendix B: MATLAB Code

This appendix lists the most important MATLAB scripts and functions required for the genetic algorithm. The main function is `VCO_main_GA3.m` and this is where the GA run is initiated. This function will call all other necessary functions for the optimization. `VCO_fitness5.m` is the function that determines the fitness of any given solution. The scripts `import_digitalCapSweep_min5mA2.m` and `S_parameter_Cleanup.m` are import scripts which do some processing so they are also included for completeness.

code/VCO_main_GA3.m

```
% Main GA Script for VCO with digital tuning caps
% Matthew Belz 2/22/2019

%[fitresult_CAPH, gof1] = fitDigCapWL(w, capL, CapEQ_high) %
    Overall cap HIGH transistor dependance
close all

%fit functions for the switched cap unit
[fitCapHigh, gof1] = fitDigCapWL2(NF, capL, CapEQ_high)
[fitCapLow, gof2] = fitDigCapWL2(NF, capL, CapEQ_low)
[fitRpON, gof3] = fitDigCapWL2(NF, capL, RP_ON_EQ)
[fitRpOFF, gof4] = fitDigCapWL2(NF, capL, RP_OFF_EQ)
[fitm, gof5] = fitDigCapWL2(NF, capL, mEQ)

% Fit functions for cross-coupled pair
[fitGMxWid_y, gof] = fitGMxWxcplY(Gm, Wxcpl);
[fitCxcplusingGm, gof] = gmXCxcplY(Gm, Cxcpl);

```

```
% [NF, CapL, varW, varNF]


lb1=[1,3.2e-6, 2.56e-7, 1]; %last element is fixed cap
ub1=[20,60e-6, 10e-6, 20];

% L for now set
% largest width and smallest outer dimension simulated
    inductance at 28G
L=130.7e-12;
L_rp= RpIND_avg; % average Rp for OD=100u, Wid=12.25u freq
    sweep 22-32GHz

opts1= optimoptions(@ga, ...
                    'PopulationSize', 200, ...
                    'MaxGenerations', 60, ...
                    'EliteCount', 2, ...
                    'ConstraintTolerance', 5, ...
                    'FunctionTolerance', 5, ...
                    'PlotFcn', @gaplotbestf);
% rng(0,'twister');
[xbest, fbest, exitflag] = ga(@(x)VCO_fitness5(x,fitCapHigh,
    fitCapLow,fitRpON,fitRpOFF,fitm,fitCV0,fitCV05,fitRP02,
    fitCV50percent,fitGMxWid_y,fitCxcplusingGm,L, L_rp), 4,
    [], [],[],[], ...
                                lb1, ub1,[] , [1,4], opts1);
```

code/VCO_fitness5.m

```
function retVal = VCO_fitness5(x,fitCapHigh,fitCapLow,fitRpON
    ,fitRpOFF,fitm,fitCV0,fitCV05,fitRP02,fitCV50percent,
    fitGMxWid_y,fitCxcplusingGm,L,L_rp)
% VCO Fitness with switched caps
% Matthew Belz 2/23/2019

% X = [tranNF, CapL, varW, varNF]

%
disp(x(1)+ "   " + x(2))
C_unit_ratio = fitm(x(1),x(2))
%unit cap C - bit b0
C_unit_high = fitCapHigh(x(1),x(2))
C_unit_low = fitCapLow(x(1),x(2))
```

```
% bit b1, 2C
C_unit_high2=fitCapHigh(x(1),x(2))*2;
C_unit_low2=fitCapLow(x(1),x(2))*2;
% Thermo bits, 4C
C_unit_high4=fitCapHigh(x(1),x(2))*4;
C_unit_low4=fitCapLow(x(1),x(2))*4;


% Rp1
Rp2=fitRpOFF(x(1),x(2))

Rp_unit=fitRpON(x(1),x(2)); % For C
Rp_4unit=fitRpON(x(1),x(2))/4; % for 4C
Rp_2unit=fitRpON(x(1), x(2))/2; %for 2C


% VARACTOR
%Measured single-ended with single varactor so I divide by 2
C_var_high = fitCV05(x(3),x(4))/2;
C_var_low = fitCV0(x(3),x(4))/2;
C_var_50percent=fitCV50percent(x(3),x(4))/2;
Rp_var=fitRP02(x(3),x(4))*2 % multiply by 2 to account for single ended

% CROSS COUPLED PAIR

%Generate worst case Rp -- All ON, 2 binary bits, 7 thermometer caps
Rp_allON=1/(1/Rp_unit+1/Rp_2unit+1/Rp_4unit+1/Rp_4unit+1/Rp_4unit+1/Rp_4unit+1/Rp_4unit+1/Rp_4unit+1/Rp_4unit +1/Rp_var+1/L_rp)
gm_needed=2/Rp_allON

% could use if statement to make sure gm is within bounds
if (gm_needed < 0.0159) && (gm_needed > 0.004924)

    Wxcpl=fitGMxWid_y(gm_needed)
    Cxcpl=fitCxcplusingGm(gm_needed)
else
    Wxcpl=0; %not sure what to put here
    Cxcpl=900e-15; % large cap
end

```

```
%Stray Cfix capacitance
Cfix=Cxcpl+10e-15; %gm pair and guess for buffer

C_mid=2*C_unit_high4+C_unit_low4*5+C_unit_high+C_unit_high2+C_var_50percent+Cfix; % middle frequency will be at half cap, mid curve
C_mid_minus1=1*C_unit_high4+6*C_unit_low4+C_unit_high2+C_unit_high+C_var_50percent+Cfix; % for testing overlap, at middle of curve above middle
C_mid_max= 2*C_unit_high4+C_unit_low4*5+C_unit_low+C_unit_low2+C_var_low+Cfix; % Max frequency at mid course tuning curve

C_min = 7*C_unit_low4+C_unit_low2+C_unit_low+C_var_low+Cfix; % Max curve, and max on the curve (varactor low)
C_min_minus1_mid = 7*C_unit_low4+C_unit_low2+C_unit_high+C_var_50percent+Cfix; % Middle of the curve on max -1 curve
C_min_minus1_low=  7*C_unit_low4+C_unit_low2+C_unit_high+C_var_low+Cfix;
C_min_high = 7*C_unit_low4+C_unit_low2+C_unit_low+C_var_high+Cfix; % Max curve, lowest frequency (max cap on that curve)
C_min_mid = 7*C_unit_low4+C_unit_low2+C_unit_low+C_var_50percent+Cfix; % Max curve, middle frequency


% Printing the C units for calculation
disp("C_unit_low4 " + C_unit_low4)
disp("C_unit_low2 " + C_unit_low2)
disp("C_unit_low " + C_unit_low)
disp("C_var_low " + C_var_low)
disp("C_min " + C_min)
% both binary are off, add fixed cap
% 01111 in binary is 2^4 or 15, half of 2^5 or 31
% T7 T6 T5 T4 T3 T2 T1 T0 | B1 B0
% 0 0 0 0 0 1 1 | 1 1

%C_mid_minus1 is 14 or 0011| 10
%
%0 0 0 0 1 1 1 | 1 0
% Low cap is  0 0 0 0 0 0 0 | 0 0
```

```
% C_min=C_unit_low+C_unit_low2+7*C_unit_low4+Cfix+C_var_low;

C_max=C_unit_high+C_unit_high2+7*C_unit_high4+C_var_high+Cfix % all caps ON
C_max_mid=C_unit_high+C_unit_high2+7*C_unit_high4+C_var_50percent+Cfix;


f_min=1/(2*pi*sqrt(L*C_max)) % absolute min f, all caps OFF, varactor high
f_min_mid=1/(2*pi*sqrt(L*C_max_mid)) %Middle of the lowest curve

f_mid=1/(2*pi*sqrt(L*C_mid)) % middle of middle curve
f_max_high=1/(2*pi*sqrt(L*C_min)) % absolute max f = all caps off (and varactor low)
f_max_low=1/(2*pi*sqrt(L*C_min_high)) % Highest curve, lowest freq
f_max_minus1 = 1/(2*pi*sqrt(L*C_min_minus1_low))% highest point on second highest curve
f_max_mid = 1/(2*pi*sqrt(L*C_min_mid)) % Highest curve, middle frequency
overlap= (f_max_minus1-f_max_low)/((f_max_high-f_max_low))


TR=f_max_mid-f_min_mid




% Find the overlap between curves
% using the mid curve


% Weights for "normalized" fitness function
alpha=1e3;
beta=1e-5;
gamma=1E-5;

retVal = alpha*abs(3-C_unit_ratio) +  beta*1/Rp2 + 1E4*1/Rp_var + gamma*abs(28e9-f_mid) + 1e6*abs(0.4-overlap)  + 1E-6*1/TR
%alpha*abs(3-C_unit_ratio)+beta*1/Rp1
```

```
end
```

code/import_digitalCapSweep_min5mA2.m

```
% imports digital switched cap sweep varibles and processes the data
% Matthew Belz 2/20/2019



% Import Sweeps
m = importMOM2('m_swCap_60u.csv', 2, 102);
caphigh = importMOM2('CapHigh_60u.csv', 2, 102);
caplow = importMOM2('CapLow_60u.csv', 2, 102);
RpON = importMOM2('Rp_ON_swCap_60u.csv', 2, 102);
Rp2_OFF = importMOM2('Rp_OFF_swCap_60u.csv', 2, 102);


capL=caphigh(:,1);
NF=[1:20];


% For m (Con/Coff) ratio
count2=1;
for k = 2:2:40 %Import as just cap values without capL columns
    mEQ(:,count2)=m(:,k);
    count2=count2+1;

end

% For high value of cap
count2=1;
for k = 2:2:40 %Import as just cap values without capL columns
    CapEQ_high(:,count2)=caphigh(:,k);
    count2=count2+1;

end

% For low value of cap
count2=1;
```

```
for k = 2:2:40 %Import as just cap values without capL columns
    CapEQ_low(:,count2)=caplow(:,k);
    count2=count2+1;

end

% For Rp1 (ON)
count2=1;
for k = 2:2:40 %Import as just cap values without capL columns
    RP_ON_EQ(:,count2)=RpON(:,k);
    count2=count2+1;

end

%For RP2 (OFF)
count2=1;
for k = 2:2:40 %Import as just cap values without capL columns
    RP_OFF_EQ(:,count2)=Rp2_OFF(:,k);
    count2=count2+1;

end




%Generate cap Width sweep
count=1;
for i=1.22e-6:0.68e-6:6.66e-6
    w(count)=i;
    count=count+1;
end

% Import the curves for the gm
GmVsWxcpl = importGmPlot('gm_wxcpl_5mA.csv', 2, 38);
CxcplVsGm = importGmPlot('gm_cin_5mA.csv',2,38);

%Generate vectors used in Curve fit
Wxcpl=GmVsWxcpl(:,1);
Gm=GmVsWxcpl(:,2);

```

```
GmC=CxcplVsGm(:,1);
Cxcpl=CxcplVsGm(:,2);



% GmVsWxcpl = importGmPlot('GmVsWxcpl2.csv', 2, 38);
% CxcplVsGm = importGmPlot('CxcplVsGm2.csv',2,38);

%Generate vectors used in Curve fit
Wxcpl2=GmVsWxcpl(:,1);
Gm2=GmVsWxcpl(:,2);

GmC2=CxcplVsGm(:,1);
Cxcpl2=CxcplVsGm(:,2);
```

code/S_Parameter_Cleanup.m

```
% S-Parameter Clean-up

sp28GHzW256nto25uNF1to25Lmin(1,:) = [];
sp28GHzW256nto25uNF1to25Lmin(:,4:3:end) = [];
sp28_new = zeros(31,ceil(length(sp28GHzW256nto25uNF1to25Lmin)/2));
sp28_new(:,1) = sp28GHzW256nto25uNF1to25Lmin(:,1);
sp28_new(:,2:end) = sp28GHzW256nto25uNF1to25Lmin(:,2:2:end) + ...
    1i*sp28GHzW256nto25uNF1to25Lmin(:,3:2:end);
voltage = sp28_new(:,1);
s11 = sp28_new(:,2:end);
sp28_new_y_partial = s11(:,1:249);
width=[256e-9,300e-9:100e-9:25e-6];
y11 = (1/50)*(1-s11)./(1+s11);
fc = 28e9;
cap = imag(y11)./(2*pi*fc);
cap_partial = cap(:,1:25);
NF = 1:1:25;
cap_super_partial = cap_partial(1:31);
matrix_cap_super = [voltage'; cap_super_partial];
cap_super_partial2 = 1e16*cap_super_partial;
matrix_cap_super2 = [voltage'; cap_super_partial2];
NF2=1:1:25;


cap_reform = reshape(cap,31,25,249);
```

```
V0=cap_reform(7,:,:); % Not at zero V, at -0.2V for most linear
V0sq=squeeze(V0);
Vw=cap_reform(7,1,:);

V05=cap_reform(13,:,:); %selects V=0.1V
V05sq=squeeze(V05);

V50percent=cap_reform(10,:,:); %selects V=-0.05v for 50% overlap measurement
V50percentSq=squeeze(V50percent);

Rp=1./(real(y11));
Rp_reform=reshape(Rp,31,25,249);
Rp02=Rp_reform(15,:,:); %Get Rp at 0.2V
Rp02sq=squeeze(Rp02);

surf(width',voltage',squeeze(cap_reform(:,1,:)), 'FaceColor', 'r')

%CO(:,:,1) = zeros(31,99); % red
%CO(:,:,2) = ones(31,99).*repmat(linspace(0.5,0.6,99),31,1); % green
%CO(:,:,3) = ones(31,99).*repmat(linspace(0,1,99),31,1); % blue

CO(:,1) = zeros(1,25); % red
CO(:,2) = ones(1,25).*linspace(0.5,0.6,25); % green
CO(:,3) = ones(1,25).*linspace(0,1,25); % blue

CO = colorSpectrum(25);

for NF=1:25
    surf(width(1:99)*1e6',voltage',squeeze(cap_reform(:,NF,1:99))*1e15,'FaceColor', CO(NF,:) ,...
        'EdgeColor', 'k', 'LineStyle', ':')
    hold on
end
xlabel('Width (\mu m)');
ylabel('Voltage (V)');
zlabel('Capacitance (fF)');
```

```
%Generate surface for V=0V
[fitCV0, gof] = createFitV0(width, NF2, V0sq);
%Generate function for voltage at V=-0.2
[fitCV05, gof] = createFitCV05(width, NF2, V05sq);
%Generate function for voltage at V=0.1;
[fitCV50percent, gof] = createFitCV05(width, NF2,
    V50percentSq);
%Generate function for voltage at V=-0.05;
[fitRP02, gof] = createFitRP02(width, NF2, Rp02sq)
%Generate function for voltage at V=0.5p;
```

Bibliography

[1] S. Elabd, S. Balasubramanian, Q. Wu, T. Quach, A. Mattamana, and W. Khalil, “Analytical and Experimental Study of Wide Tuning Range mm-Wave CMOS LC-VCOs,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 5, pp. 1343–1354, May 2014.

[2] L. Simic, S. Panda, J. Riihijarvi, and P. Mahonen, “Coverage and Robustness of mm-Wave Urban Cellular Networks: Multi-Frequency HetNets Are the 5g Future,” in *2017 14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Jun. 2017, pp. 1–9.

[3] S.-S. Song and H. Shin, “An RF model of the accumulation-mode MOS varactor valid in both accumulation and depletion regions,” *IEEE Transactions on Electron Devices*, vol. 50, no. 9, pp. 1997–1999, Sep. 2003.

[4] R. Pvoa, I. Bastos, N. Loureno, and N. Horta, “Automatic synthesis of RF front-end blocks using multi-objective evolutionary techniques,” *Integration, the VLSI Journal*, vol. 52, pp. 243–252, Jan. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167926015000462

[5] S. Alzahrani, S. Elabd, and W. Khalil, “A Wide Tuning Range Millimeter Wave CMOS LCVCO with Linearized Coarse Tuning Characteristics,” in *2018 16th IEEE International New Circuits and Systems Conference (NEWCAS)*. Montral, QC, Canada: IEEE, Jun. 2018, pp. 78–82. [Online]. Available: https://ieeexplore.ieee.org/document/8585562/

[6] R. Gonzlez-Echevarra, E. Roca, R. Castro-Lpez, F. V. Fernndez, J. Sieiro, J. M. Lpez-Villegas, and N. Vidal, “An Automated Design Methodology of RF Circuits by Using Pareto-Optimal Fronts of EM-Simulated Inductors,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 15–26, Jan. 2017.

[7] S. Levantino, C. Samori, A. Zanchi, and A. L. Lacaita, “AM-to-PM conversion in varactor-tuned oscillators,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 7, pp. 509–513, Jul. 2002.

[8] Q. Wu, T. Quach, A. Mattamana, S. Elabd, S. R. Dooley, J. J. McCue, P. L. Orlando, G. L. Creech, and W. Khalil, "A 10mw 37.8ghz current-redistribution BiCMOS VCO with an average FOMT of 193.5dbc/Hz," in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, Feb. 2013, pp. 150–151.

[9] S. Elabd and W. Khalil, "Impact of technology scaling on the tuning range and phase noise of mm-wave CMOS LC-VCOs," *Integration*, vol. 52, pp. 195–207, Jan. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167926015000796

[10] K. A. De Jong, "Genetic Algorithms Are NOT Function Optimizers," in *Foundations of Genetic Algorithms*. Elsevier, 1993, vol. 2, pp. 5–17. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/B9780080948324500064

[11] R. A. d. L. Moreto, C. E. Thomaz, and S. P. Gimenez, "Gaussian Fitness Functions for Optimizing Analog CMOS Integrated Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 10, pp. 1620–1632, Oct. 2017.

[12] B. Sadhu, M. A. Ferriss, A. S. Natarajan, S. Yaldiz, J. Plouchart, A. V. Rylyakov, A. Valdes-Garcia, B. D. Parker, A. Babakhani, S. Reynolds, X. Li, L. Pileggi, R. Harjani, J. A. Tierno, and D. Friedman, "Correction to A Linearized, Low Phase Noise VCO Based 25 GHz PLL With Autonomic Biasing," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 6, pp. 1539–1539, Jun. 2013.

[13] M. Ferriss, A. Rylyakov, J. A. Tierno, H. Ainspan, and D. J. Friedman, "A 28 GHz Hybrid PLL in 32 nm SOI CMOS," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 4, pp. 1027–1035, Apr. 2014.

[14] J. F. Osorio, C. S. Vaucher, B. Huff, E. v. d. Heijden, and A. d. Graauw, "A 21.7-to-27.8ghz 2.6-degrees-rms 40mw frequency synthesizer in 45nm CMOS for mm-Wave communication applications," in *2011 IEEE International Solid-State Circuits Conference*, Feb. 2011, pp. 278–280.

[15] Q. Wu, T. K. Quach, A. Mattamana, S. Elabd, P. L. Orlando, S. R. Dooley, J. J. McCue, G. L. Creech, and W. Khalil, "Frequency Tuning Range Extension in LC-VCOs Using Negative-Capacitance Circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 4, pp. 182–186, Apr. 2013.

[16] E. Mammei, E. Monaco, A. Mazzanti, and F. Svelto, "A 33.6-to-46.2ghz 32nm CMOS VCO with 177.5dbc/Hz minimum noise FOM using inductor splitting for tuning extension," in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, Feb. 2013, pp. 350–351.

[17] E. Chang, J. Han, W. Bae, Z. Wang, N. Narevsky, B. NikoliC, and E. Alon, "BAG2: A process-portable framework for generator-based AMS circuit design," in *2018 IEEE Custom Integrated Circuits Conference (CICC)*, Apr. 2018, pp. 1–8.

[18] J. J. Rael and A. A. Abidi, "Physical processes of phase noise in differential LC oscillators," in *Proceedings of the IEEE 2000 Custom Integrated Circuits Conference (Cat. No.00CH37044)*, May 2000, pp. 569–572.