Fast Design Exploration of Nanoscale Circuits

Designer's Guide

Saraju P. Mohanty and Elias Kougnianos
NanoSystem Design Laboratory (NSDL, `http://nsdl.cse.unt.edu/`)
University of North Texas, Denton, TX – 76207, USA.

1 Introduction

A phase-locked loop (PLL) is a typical typical mixed-signal system which is a part of every synchronous circuit or system that needs a clock [Gar12]. This guide is presents the fast design methods using this important circuit as a case study. This manual has shown the steps that are needed to produce optimal values for PLL design. After the optimal design parameters are identified by the optimization algorithm the designer is left to perform final adjustments to the physical layout that was created previously.

The design flow proposed in this is shown in Fig. 1 [Gar12, GMK12b]. As the input specifications are received by the designer the initial logical design is created. The physical layout is created based on this design when it meets specifications. The usual technology rule checks (DRC), layout versus schematic (LVS) and parasitic extraction (RCLK) steps are then performed. The RCLK components have tremendous effect on high frequency AMS circuits. As most of the time the circuit fails specifications, extra steps are introduced to bring the design back to spec without iteratively going back to the physical layout or logical design. It may be noted that the figure shown above is for polynomial metamodels and bee colony optimization; however other types of metamodels and optimization algorithms can also be used in a similar design optimization flow.

2 A Mixed-Signal Case Study Circuits: Phase Locked Loop (PLL)

In day to day life the circuit or systems that are used are synchronous systems or circuits. There is a global clock which is understood and followed by all portions of the synchronous circuits and systems. The clock signals are generated by a PLL. The PLL is a device with many interesting applications, including frequency synthesis, frequency demodulation, synchronization, tracking, ranging and television sweep circuits. A PLL is used in fundamentally three different ways as discussed below:

- The PLL is used as a demodulator. The phase-locked loop may be thought of as a matched filter operating as a coherent detector.
- The PLL is used as a tracker of a carrier. It may be thought of as a narrow-band filter for removing noise from the signal and regenerating a clean replica of the signal.
- The PLL is used as a frequency synthesizer, where an oscillator is locked to a multiple of an accurate reference frequency. A voltage controlled oscillator (VCO) is divided down to a reference frequency that is locked to a frequency derived from an accurate source such as a crystal oscillator.

This motivates to use PLL as a case study mixed-signa system. The PLL consisting of analog and digital components makes a fitting example of a mixed-signal system [GMK12b]. Each of the components of a PLL needs specific attention in design and optimization. At the same time there are global figures-of-merit which are common to every component as well as the PLL.

3 Phase Locked Loop (PLL)System Overview

The Phase locked loop (PLL) system is a closed loop feedback control system that consists of the Phase Detector(PD), Charge Pump (CP), LC Voltage Controlled Oscillator (LC-VCO) and Frequency Divider (FD). It provides a good example of a mixed-signal system. The reduction of phase difference between a locally generated signal and a reference signal is the basic idea behind the working of a PLL [Wan05b, GMK12b].

The main feature of the PLL is that it maintains a generated signal in a fixed phase relationship to a reference signal. The phase frequency detector detects the difference in phase or frequency between the reference

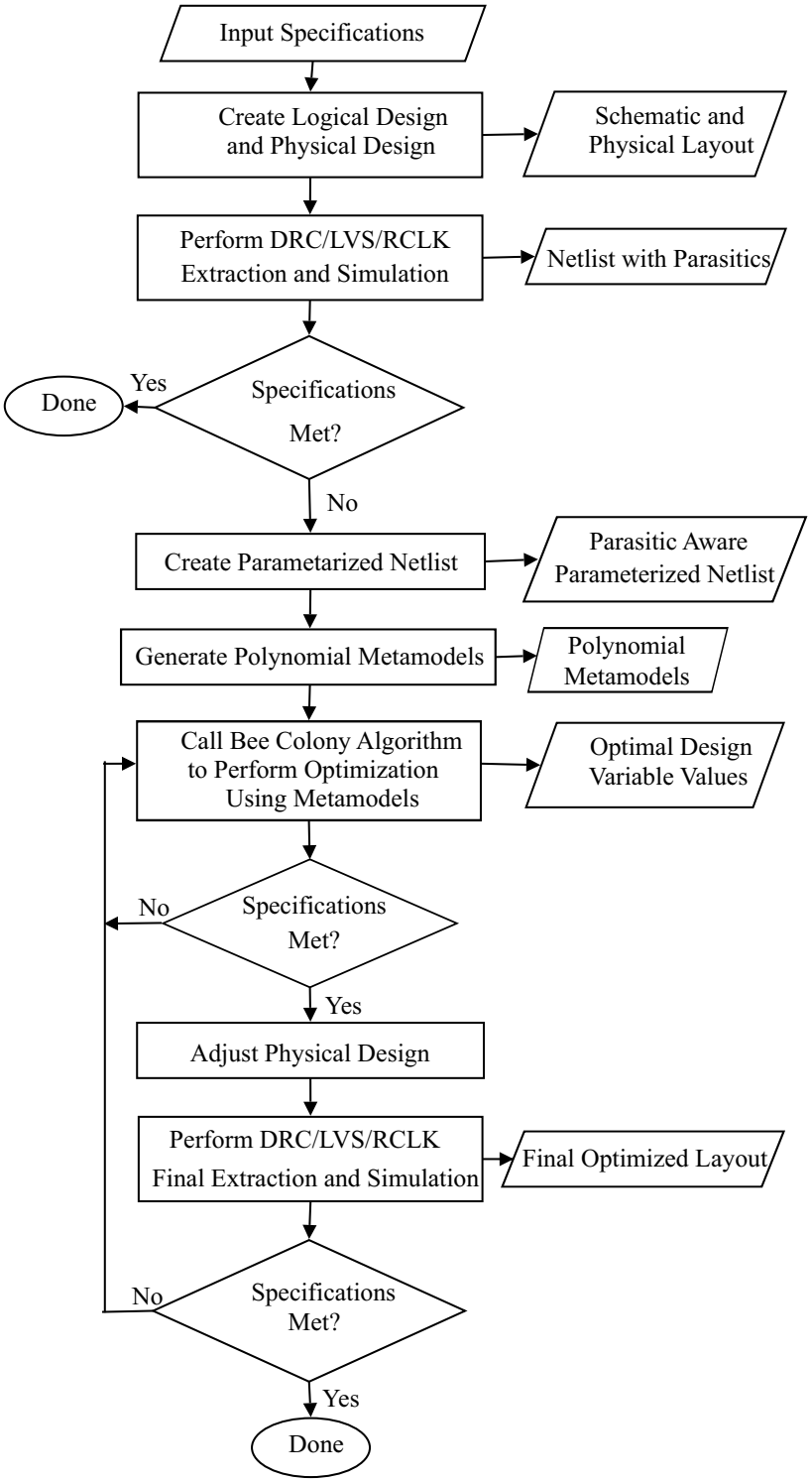


Figure 1: The proposed ultra-fast design optimization flow that uses polynomial metamodels and bee colony optimization.

signal and the feedback signal and produces an output signal proportional to the difference in phase between the two signals. This output signal of the phase detector goes through the charge pump which supplies the current to the low pass loop filter. As a result, the control voltage of the voltage controlled oscillator is adjusted so as to oscillate according to the control voltage at its input. Depending upon the frequency of the reference signals, the frequency of the output signal from the LC-VCO is divided using a frequency divider. The output of the frequency divider is fed back to the phase frequency detector to get back to the reference frequency range. In other words, the PLL can be used as a frequency multiplier that stays in phase with the reference signal. It is common practice to use the divider in powers of two due to their ease of implementation. When the loop is locked, the frequency of the LC-VCO is exactly equal to the average frequency of the input signal [Gup75]. Therefore, a PLL responds both to the frequency and phase of the input signals by automatically raising or lowering the frequency of the LC-VCO until it is matched to the reference in both frequency and phase.

The PLL is designed and simulated in many ways and at different levels; for example, behavioral simulation or circuit level simulation. In the following sections the detailed properties of individual components are discussed. The component wise schematic design is performed which is then merged to obtain a flat schematic design of the PLL. In the next level, layout design (physical design) of the individual components is performed which are then merged to perform the overall physical design of the PLL. The working of sub-components of the PLL system is discussed further in the following sections.

Some design metrics need to be known for the components of a PLL to begin the design. This is essential to ensure that the design meets the target specification. The feedback loop transfer function for a PLL is calculated as follows [BLB98, Gar12]:

$$H(s) = \left(\frac{K_{pdi} K_{VCO} (1 + sRC_1)}{s^2 + \frac{K_{pdi} K_{vco} R}{N} + \frac{K_{pdi} K_{VCO}}{NC_1}} \right). \tag{1}$$

The natural frequency is derived from the transfer function presented as follows:

$$\omega_n = \left(\sqrt{\frac{K_{pdi} K_{VCO}}{NC_1}} \right). \tag{2}$$

The damping factor is calculated using the following formula:

$$\xi = \left(\frac{\omega_n}{2} RC_1 \right). \tag{3}$$

The damping factor was set to $\xi = 1$. The pull in time can be calculated as follows:

$$T_p = \left(2RC_1 \ln \frac{\frac{K_{VCO}}{N} I_{pump}}{s^2 + \frac{K_{pdi} K_{VCO} R}{N} + \frac{K_{pdi} K_{VCO}}{NC_1}} \right). \tag{4}$$

The above is based on the VCO's tuning range. By substituting the above formulas, the natural frequency of a VCO is rewritten in the following manner:

$$\omega_n = \left(\sqrt{\frac{\frac{I_{pump}}{2\pi} K_{VCO}}{2C_1}} \right). \tag{5}$$

The locking range of the PLL is determined using the following formula:

$$\omega_L = 4\pi\xi\omega_N. \tag{6}$$

4 Phase Locked Loop (PLL) Components

4.1 Phase Detector

The phase detector is the core element of a phase locked loop. Its action enables the phase differences in the loop to be detected and the resultant error voltage to be produced. A phase detector directs the charge pump to supply charge amounts proportional to the phase error detected. The range of the phase detector varies from a very simple XOR gate to complex logic circuit consisting of flip-flops. The phase detector is an analog mixer [Wan05a]. It also functions as an asynchronous sequential logic circuit so as to detect mismatch between phase and frequency between two signals. The phase detector is used for better accuracy at small phase differences and to lock phase signals at high frequencies. The diagram of the implemented phase detector using two D flip-flops and one AND gate is shown in Fig. 2. The gain of the phase detector is calculated using the following expression:

$$K_{tri} = \left(\frac{V_{dd}}{4\pi}\right). \tag{7}$$

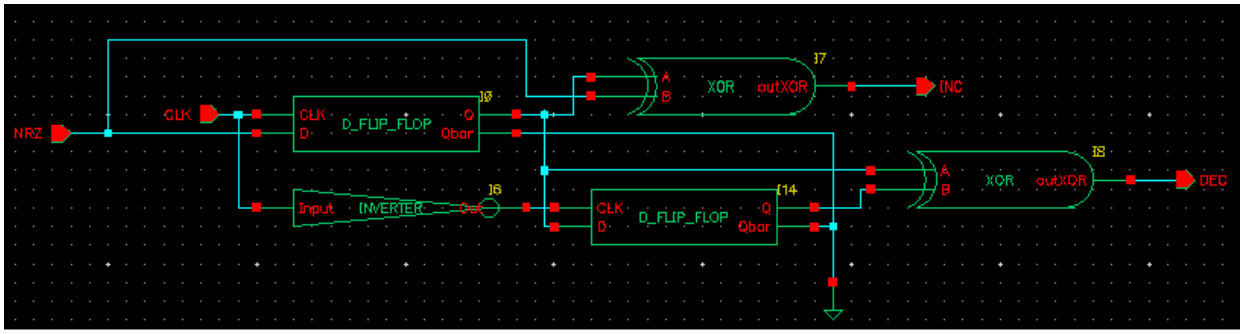


Figure 2: Phase detector.

4.2 Charge Pump and Loop Filter

The charge pump transistor level schematic is shown in Fig. 3. The charge pump plays a very important role along with the other components in the PLL. The stabilization of spurious fluctuations of currents and switching time to minimize the spurs in the VCO input is acquired by the charge pump. An efficient charge-pump circuit is one which can achieve high voltage from the available low supply voltage. These pumps manipulate the amount of charge on the filters of the capacitors depending upon the signals from the up and down of the phase frequency detector. Charge pumps are essentially utilized to convert timed logic levels into analog quantities for controlling the locked oscillators [Gar80]. The gain for the charge pump is calculated using the following expression:

$$K_{pdi} = \left(\frac{I_{pump}}{2\pi}\right). \tag{8}$$

Filter circuits are used in a wide variety of applications. In the case of a PLL, the loop filter smooths the phase difference output from the phase detector for input to the VCO. The output signal from the charge pump is applied to the loop filter. The loop filter determines the dynamic characteristics of the PLL. A low-pass RC filter is used in the design presented in this dissertation to pass frequency signals within the range of the VCO. The cutoff frequency of the loop filter should be approximately equal to the maximum frequency

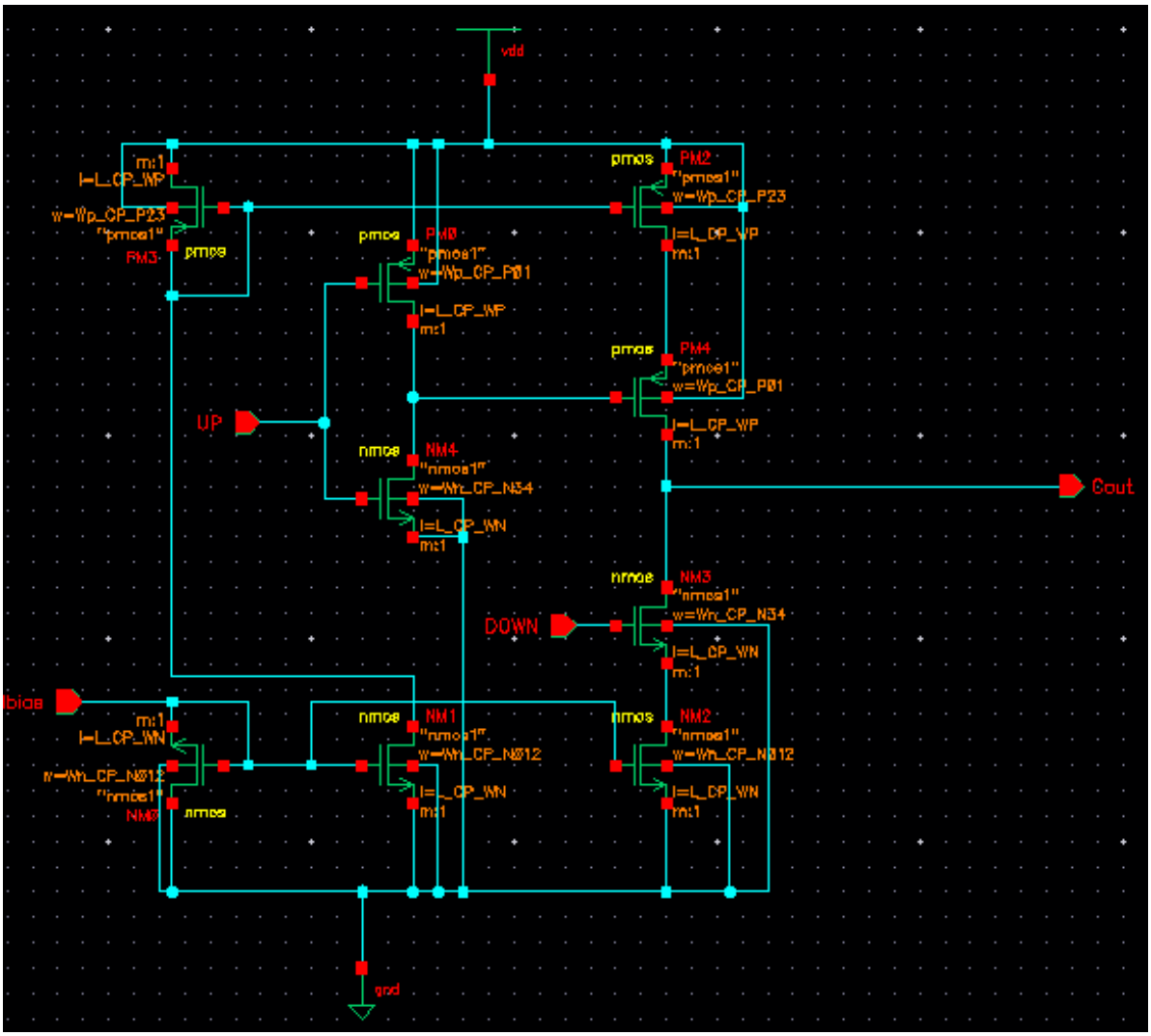


Figure 3: Schematic diagram of the charge pump.

of the LC-VCO. This enables the filter to reject signals at frequencies above the maximum frequency of the VCO. The parasitic values for the loop filter are determined based on its cutoff frequency. The loop filter is designed to match the characteristics required by the application of the PLL. The loop filter can affect tracking and capture ranges and maximum slew rate.

Capacitor C_2 prevents the charge pump voltage from causing voltage jumps on the input of the LC-VCO and thus frequency jumps in the PLL output. For slow variations in the phase, the current, I_{pump}, linearly charges C_1 and C_2. This gives an average effect and therefore small deviation in V_{in}. For fast variations in phase, the charge pump simply drives the resistor R, eliminating the averaging and allowing the VCO to track quickly moving variations in the input data. In general C_2 is set to roughly one tenth of C_1. Due to such low capacitance, the capacitance of C_2 can be neglected. The Loop Filter transfer function is given by the following expression:

$$K_f = \left(\frac{1 + sRC_1}{sC_1} \right). \tag{9}$$

The filter resistance and capacitance can be determined by the following expression:

$$RC_1 = \left(\frac{2}{\omega_N}\right). \tag{10}$$

The capacitors and resistors of the loop filter were calculated to be C_1 = 10 pF C_2 = 1 pF and R = 1.7 KΩ for the present LC-VCO specifications.

4.3 LC-VCO

An Inductor-capacitor (LC) tank based oscillator or LC voltage-controlled oscillator (LC-VCO) is an electronic oscillator specifically designed to be controlled in oscillation frequency [GMK12a]. The LC-VCO is mainly controlled by applying a DC input voltage by the loop filter. The LC-VCO usually produces cleaner output, but captures significantly more area in comparison to the ring oscillator. The gain of LC-VCO is calculated using the following expression:

$$K_{VCO} = \left(2\pi\frac{ChangeinFreq}{ChangeinVtune}\right), \tag{11}$$

$$= 2\pi\left(\frac{f_{max} - f_{min}}{V_{max} - V_{min}}\right). \tag{12}$$

Where the gain is essentially the slope of the tuning range converted to radians.

A conventional complementary NMOS and PMOS cross-coupled LC-VCO circuit is used and shown in Fig. 4. This consists of the LC-tank and several transistors. However, when the circuit is considered at the layout level with the associated parasitics the number of circuit elements in the parasitic-aware netlist is quite significant.

A well known formula for finding frequency oscillations is the following:

$$f_{osc} = \left(\frac{1}{2\pi\sqrt{L_{tank}C_{tank}}}\right). \tag{13}$$

In the above expression, L_{tank} is the inductor. The tank capacitor is calculated as follows:

$$C_{tank} = C_1 + C_2 + C_{other}. \tag{14}$$

Where C_1 and C_2 are the capacitance of the varactors, C_{other} is the summation of NMOS and PMOS gate to drain/source and other parasitic capacitances of the circuit.

The inductor tank was chosen to be as large as possible to minimize power consumption. This helped in changing the varactor capacitance and transistor sizes only. However, changing the size of transistors does not noticeably change their capacitance. Hence, an appropriate value for the varactors was first chosen, followed by an adjustment of the width of the transistors. This allowed fine tuning of the transistor sizes to the needed frequency.

As the LC-VCO has to be a symmetric circuit, all the components mirror each other to produce V_{outp} and V_{outn} with the same frequency. Therefore, the PMOS and NMOS transistors should have the same value and are denoted as parameters W_p and W_n, respectfully.

Many numerical methods provide an inaccurate representation for frequency output since they cannot account for all the parasitics of the circuit. The target frequency was chosen to be 2.5 GHz. The target frequency in the middle of the VCO's tuning range for schematic output as shown in Fig. 5. However, as shown later, the frequency dropped dramatically because of the parasitic effects.

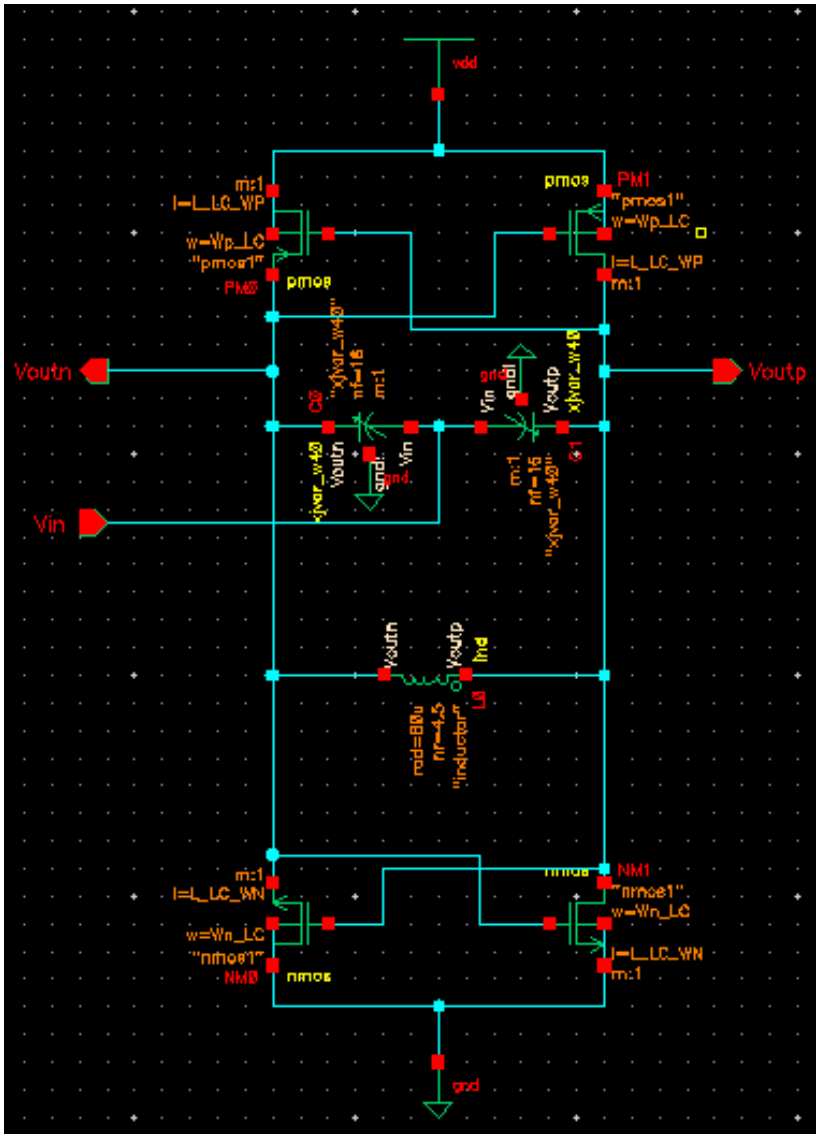


Figure 4: Schematic of the LC-VCO circuit.

5 Characterization of the Overall PLL

The Analog Design Environment is used to setup for PLL schematic simulations as shown in Fig. 6. The characterization of outputs is as follows:

settling time:
settlingTime(VT("/I3/net11") 0.2 nil ymax(VT("/I3/net11")) t 30 nil "time")

horizontal eye opening:
eyeDiagram(clip(VT("/out") 4e-07 5e-07) 4e-07 5e-07 (1 / (2 * frequency(clip(VT("/out") 4e-07 5e-07)))) ?advOptions 'horizontal)

vertical eye opening:
eyeDiagram(clip(VT("/out") 4e-07 5e-07) 4e-07 5e-07 (1 / (2 * frequency(clip(VT("/out") 4e-07 5e-07)))))

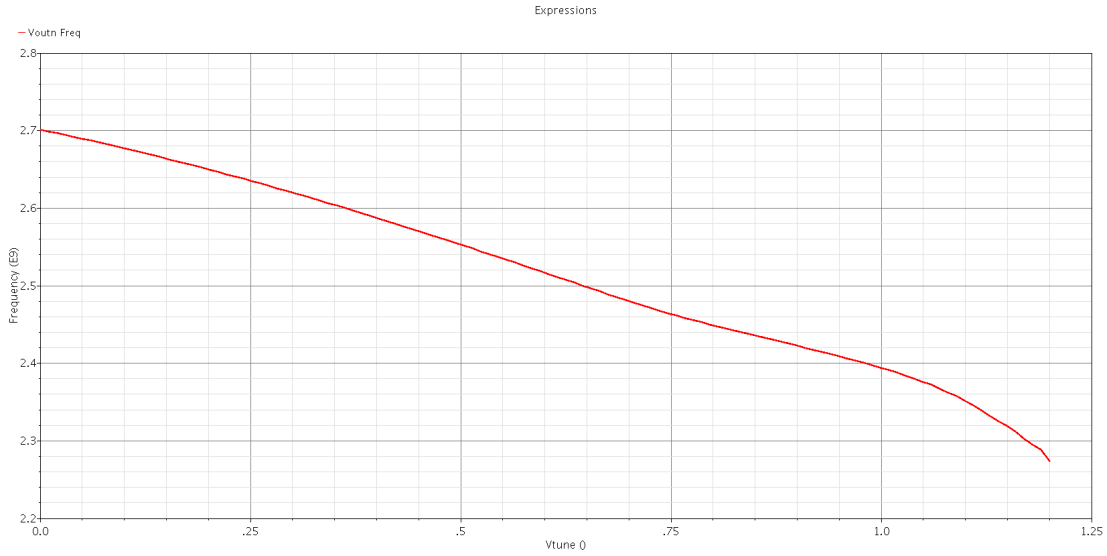


Figure 5: Measured frequency tuning characteristic of the LC-VCO (extracted using parametric analysis.)

?advOptions 'vertical)

frequency vs time diagram:
(freq VT("/out") "falling" ?xName "time" ?mode "auto" ?threshold 0.0)

average frequency value:
frequency(clip(VT("/out") 4e-07 5e-07))

average power:
average((IT("/V0/MINUS") * VT("/vdd!")))

Transient analysis setup is shown in Fig. 7. Periodic steady state analysis setup is shown in Fig. 8. Phase noise analysis is shown in Fig. 9. The phase noise is shown in Fig. 10 for the center frequency of 2.5 GHz. At 1 MHz offset from the carrier, the phase noise is -117 dBc/Hz. Due to low phase noise, we decided to focus this study on minimizing the power consumption.

6 Physical Design of the PLL Components

Once the schematic design is performed and characterized, the physical design or layout is designed. The physical layout of the LC-VCO is shown in Fig. 11. The netlist parasitic-aware netlist is obtained from this layout. The parasitic effects of the circuit were taken into consideration and the layout was made symmetrical. The symmetry of the layout provides up-conversion and the even-order distortion in the differential output waveform [H. 05]. The wire widths were maximized so as to minimize wire resistance. This also provided some space to change the specification of the transistors in future. The layout would then be recreated the second time with the final values.

The simulation on the parasitic extracted netlist, with the same sizing of the devices as in the schematic, shows the effect of the parasitics on the circuit. Since the parasitics were considered for the specific layout, it was assumed that minor modifications on few devices would not change the parasitic results drastically in the future analysis. Optimization was performed on the sizing of the devices of this circuit with extracted parasitics. Finally, the parasitics were only considered in the final layout to study its effects on the optimization calculations.

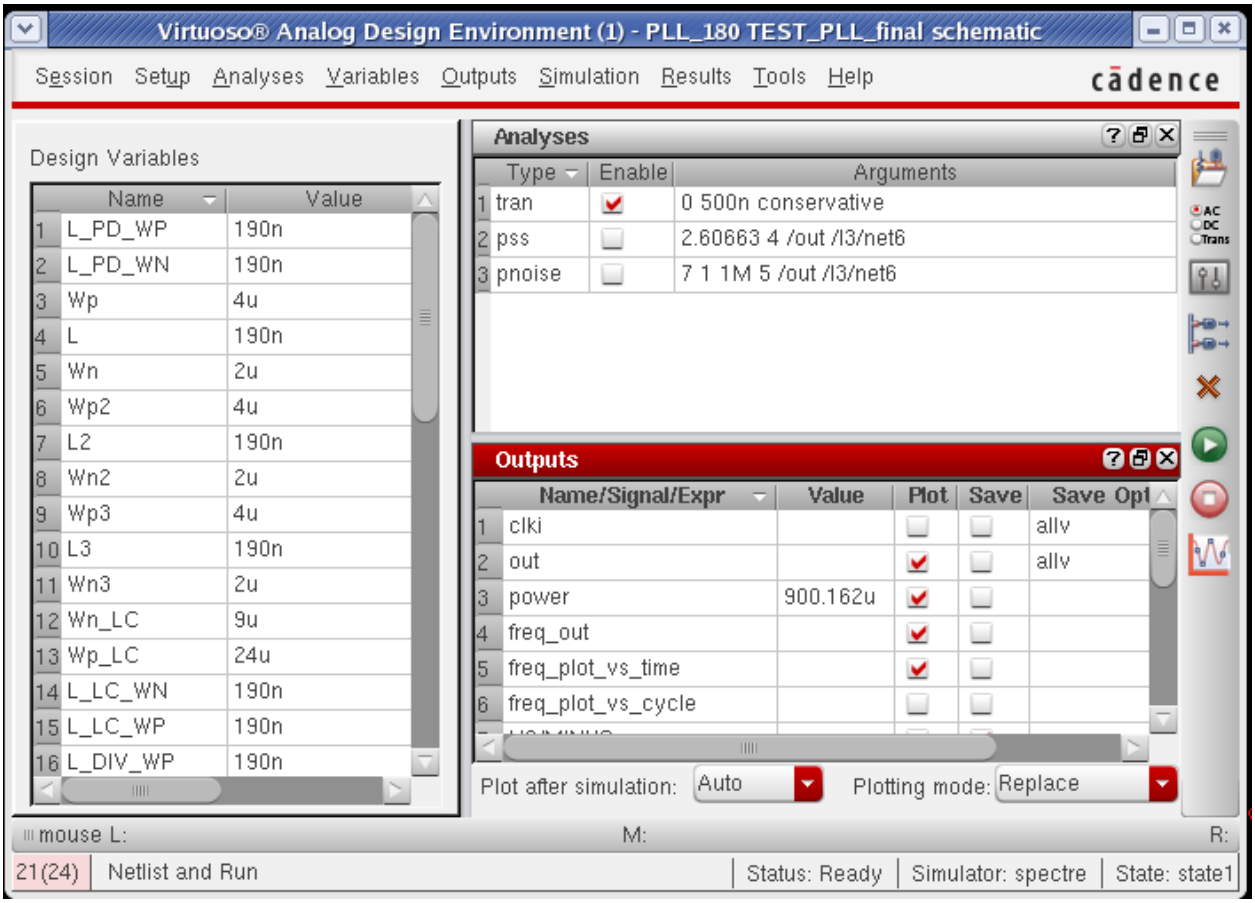


Figure 6: Virtuoso Analog Design Environment Setup.

The physical design is created for each sub-circuits of the PLL as presented in Fig. 11 [GMK12b, Gar12]. The DRC/LVS checks are conducted to verify the correctness of the layout. Final physical layout is created from stitching the above sub-circuits. The physical design is also a subject to DRS/LVS checks and parasitic (RLCK) extraction. The overall physical design for 180nm CMOS is presented in Fig. 12 [GMK12b, Gar12].

7 Netlist Parameterization for Iterative Simulation and Optimization

After a successful RLCK extraction netlist has been created the netlist needs to be edited to correspond to the schematic netlist that includes parameters [Gar12]. Please note that is very useful to use schematic net names, which is a setting in RLCK extraction. The portion of extracted netlist is broken into two portions as shown in Fig. 13. The first section is that one that includes the parasitic devices. The second section is the one that includes circuit devices. The circuit section is the only section that needs adjustment as shown in figure for PMOS device. This step requires some manual labor, or a scripting language can be used to implement this step if it is needed.

8 Design Space Sampling for Metamodel Creation

Latin hypercube sampling is used to create training samples for metamodel. The design space sampling points are generated using MATLAB. Function “lhs_design.m” takes single input (n) for number of samples.

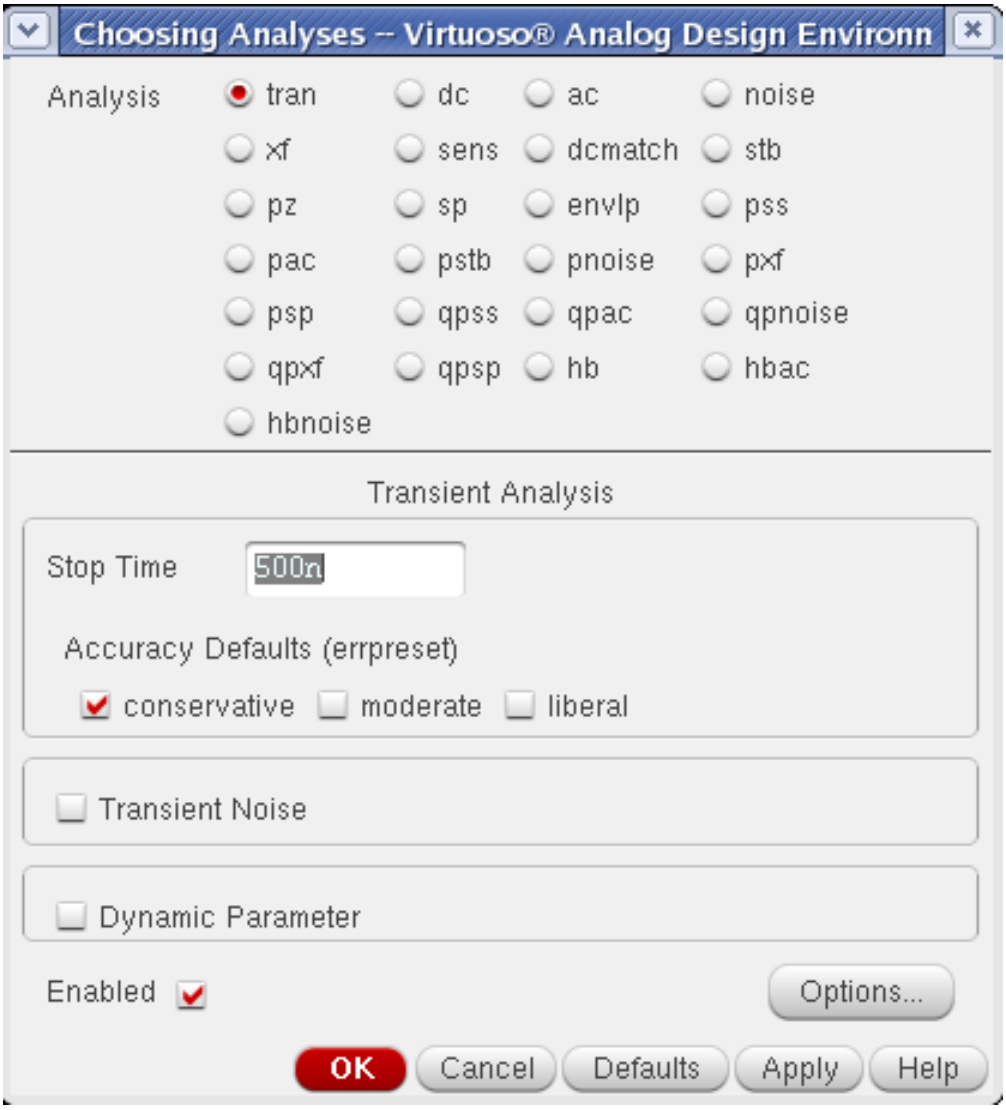


Figure 7: Transient Analysis Setup.

It is specifically designed for this circuit. The function produces a parameterized portion of ocean script creating arrays for each parameter within the strict bounds. Furthermore, the ocean file has to be adjusted to facilitate for a large number of runs. The main portion of the ocean script that is adjusted to run samples for each X matrix (LHS) row is shown in Fig. 14. The prepare portion of the ocean script is shown in Fig. 15.

The output is stored in the separate file that keeps the Y matrix output for each training set. At the completion of the simulation run, the Y matrix is then imported to MATLAB and can be used for metamodel creation. This setup allows designer to restart simulations at any specific point by adjusting the for loop parameters and easily merging the output files. Same method is used to create another smaller set for verification. Usually it needs to be roughly 30% of main training set and is used to make sure that the model is not over-fitted. As an example, design space for a 180nm CMOS LC-VCO is shown in Fig. 16 [GMK12b, GMK12a].

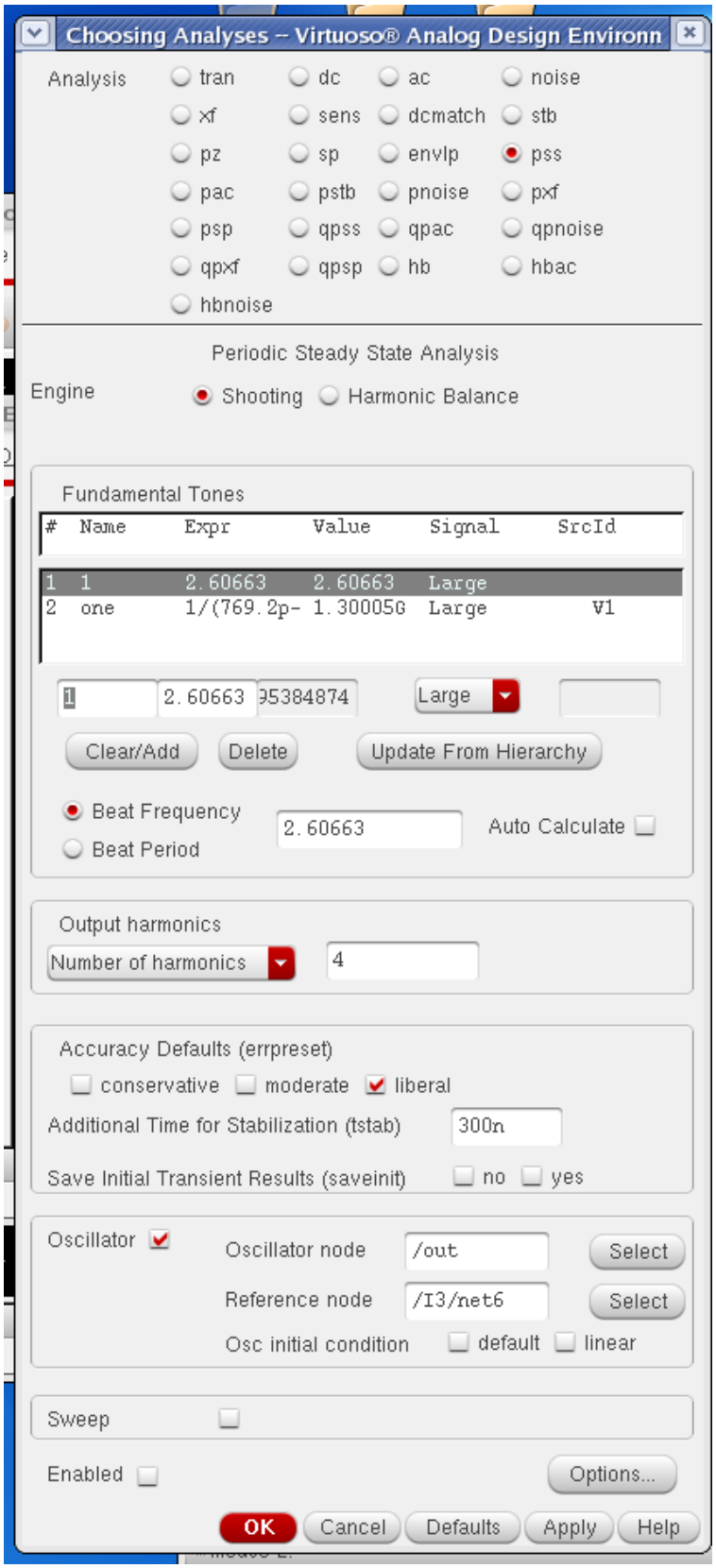


Figure 8: Periodic Steady State Analysis Setup.

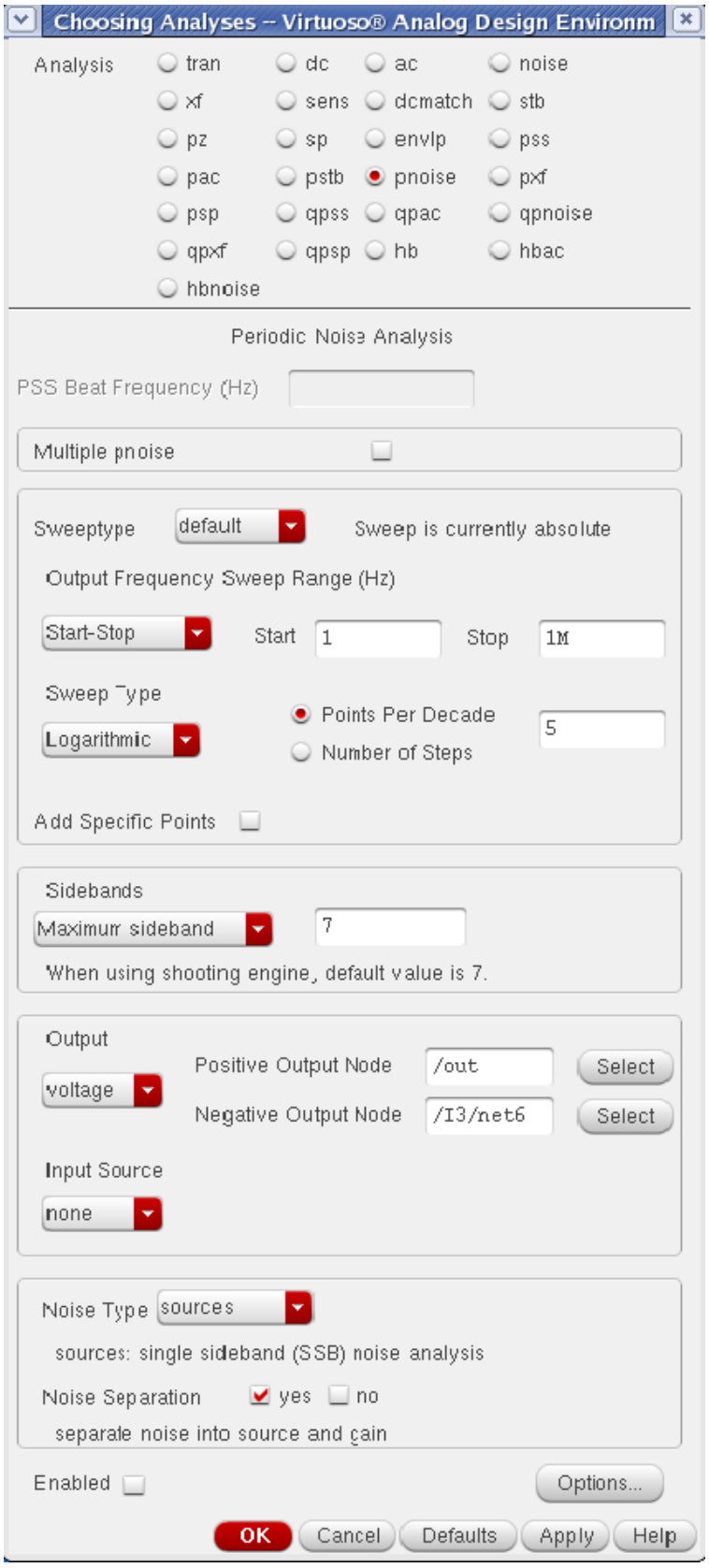


Figure 9: Phase Noise Setup.

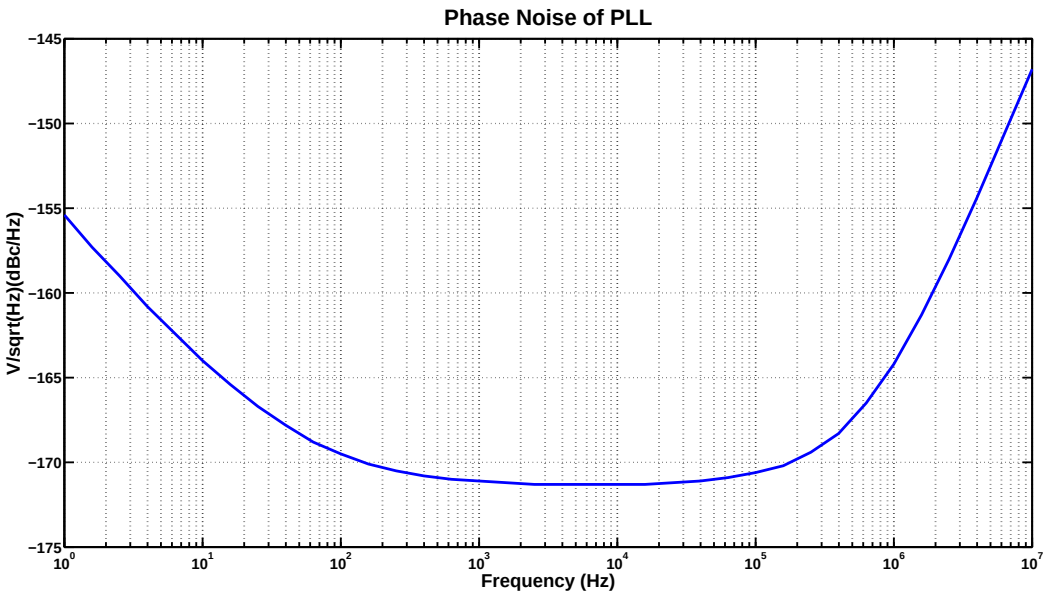


Figure 10: Phase Noise of Phase Locked Loop.

9 Metamodel Creation

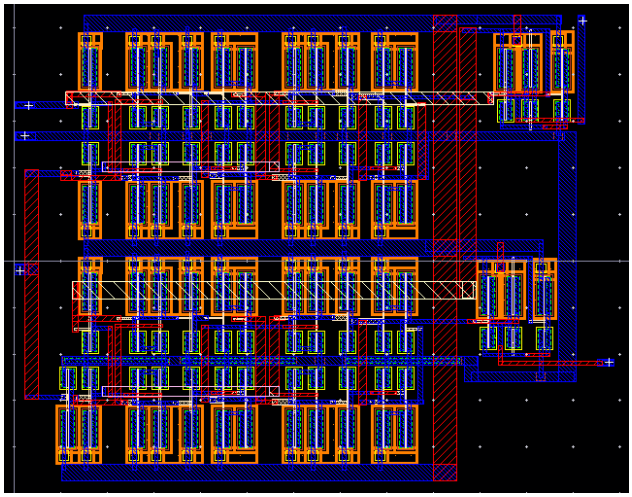
The polynomial metamodel generation steps are shown in Fig. 17 [GMK12b, GMK12a]. Other flows are followed to create non-polynomial metamodels such as artificial neural networks [GMK12c]. To maximize the accuracy of the model an appropriate sampling technique should be identified. The current paper uses Latin Hypercube Sampling (LHS) as it is accurate and yet fast. The sample data generation stage is the slowest part of the metamodeling process. The accuracy of the metamodel is dependent on the amount of simulations which is limited by the available simulation budget (time-wise). The Latin Hypercube sampling (LHS) is conducted on the number of parameters that exist in the parameterized netlist with parasitics. LHS is a semi-uniform statistical technique that divides the design space space into "Latin" squares which are then sampled randomly. The Latin squares provide more uniform distribution of samples than Monte Carlo sampling which takes random points from the design space with a given distribution for each present parameter.

In this example, the metamodel creation is conducted using Matlab. Any mathematical/statistical tool or even language/script can be used to create a metamodel. The polynomial metamodel is created as an example for simplicity. The accuracy of the metamodel is calculated by using RMSE and STD equations:

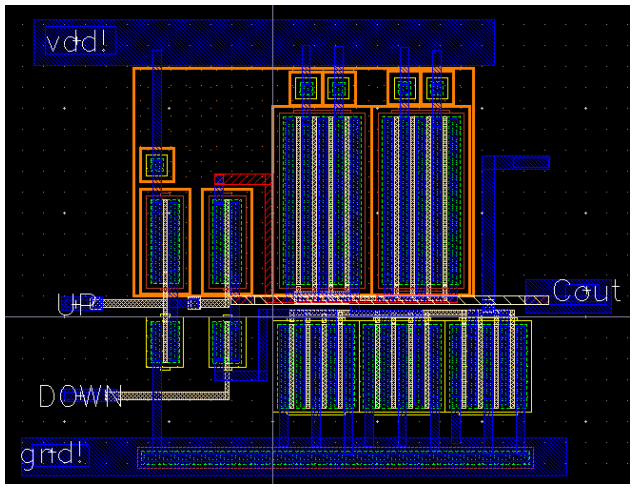
$$RMSE=\sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(y(W_{n_i},W_{p_j})-\hat{y}(W_{n_i},W_{p_j}))^2}, \tag{15}$$

$$\sigma=\sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(|y(W_{n_i},W_{p_j})-\hat{y}(W_{n_i},W_{p_j})|-RMSE)^2}. \tag{16}$$

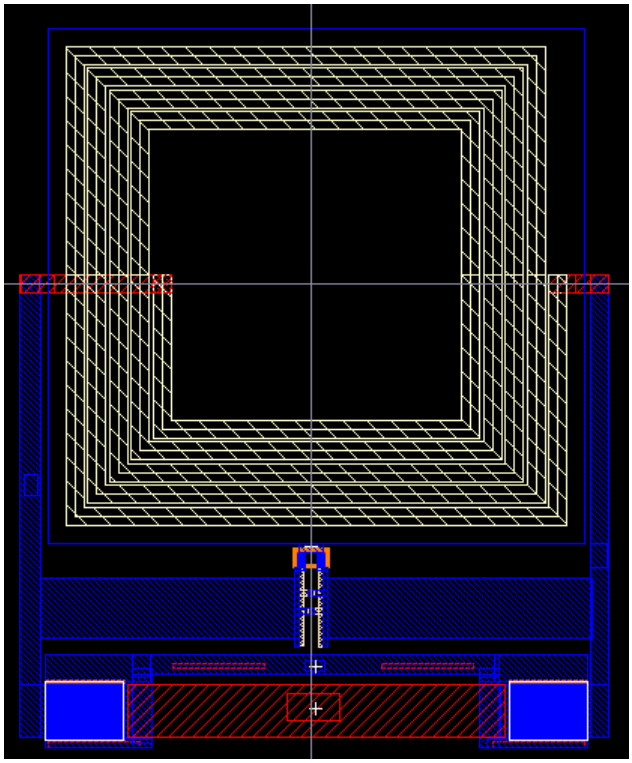
Polynomial metamodel creation up to degree of 9 can be conducted using the "metamodel" toolbox that has been developed for this research in shown in Fig. 18.



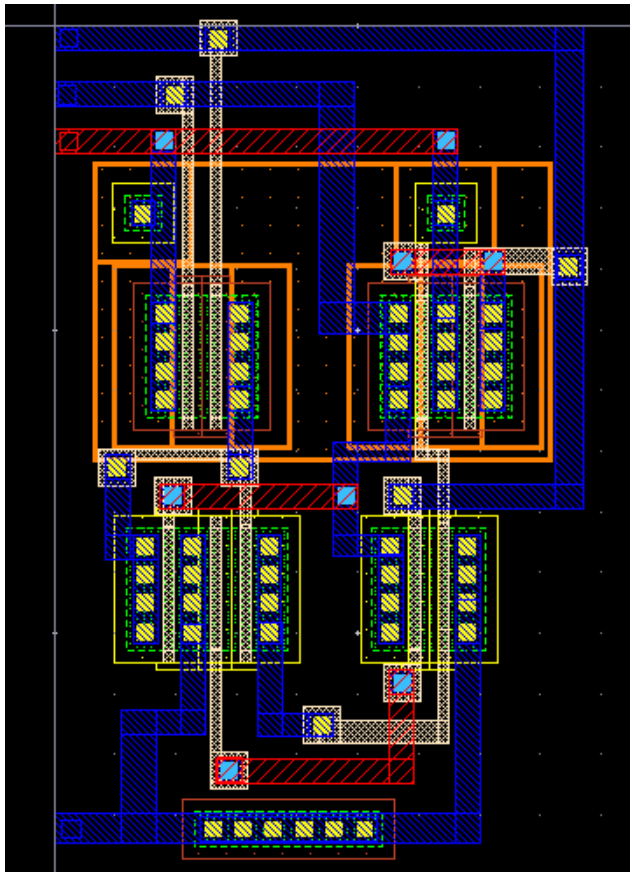
(a) Physical Layout of Phase Detector Circuit.



(b) Physical Layout of Charge Pump Circuit.



(c) Physical Layout of LC-VCO Circuit.



(d) Physical Layout of Divider Circuit.

Figure 11: Example of working physical designs of PLL components.

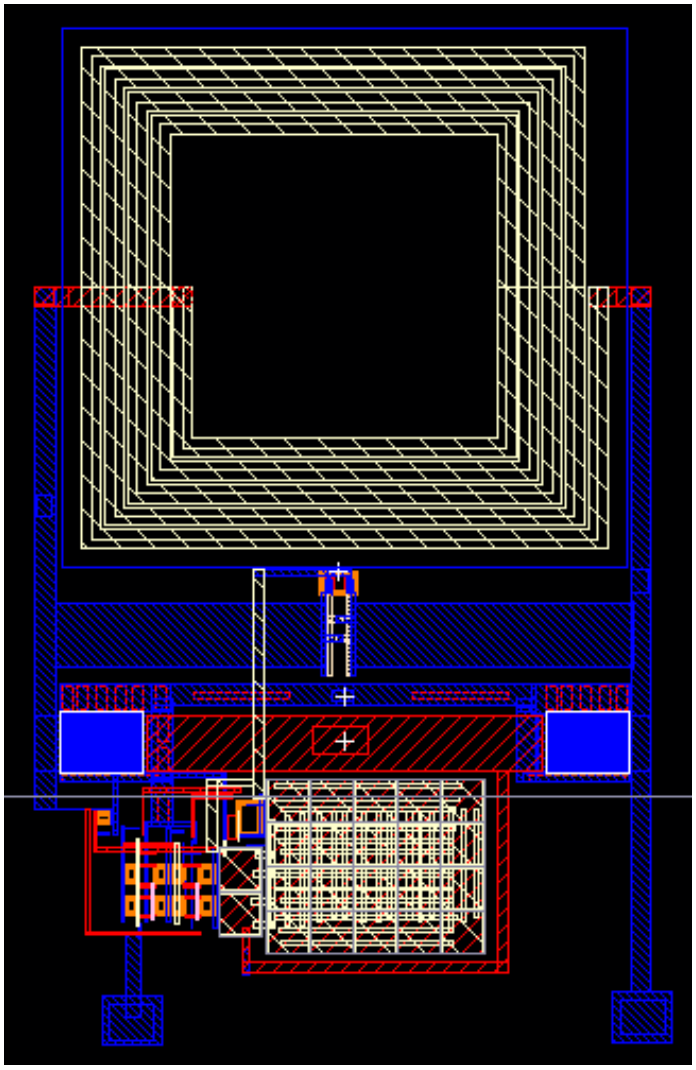


Figure 12: Physical Layout of PLL Circuit for 180nm CMOS.

In the metamodel creation section of the tool first identify X and Y matrices for metamodel generation. Pick the order of the metamodel you want to generate. Stepwise regression method is used for polynomial generation. This method can have different performance output accuracy depending on the starting position of the included coefficients. The choice to include all or to exclude all coefficients is given. Also, the choice of centering X and Y matrices is given. Press button create. The statistics for the metamodel regression is displayed on the bottom of the section. Once the metamodel of the picked order is generated it is essential to verify it with the verification data set that was created earlier. First pick X and Y values for verification data set. Then click on the Verify button and the statistical data for that metamodel is generated. Since the time to generate a metamodel is not very large, an exhaustive generation for all available orders can be performed in minutes. This allows to generate the most accurate polynomial metamodel. Please note that depending on the amount of parameters in X matrix can largely effect the order of the metamodel. The regression of highly dimensional design space requires a lot of memory space in the algorithm. For very high dimensional metamodel please look into neural network toolbox that is built in MATLAB.

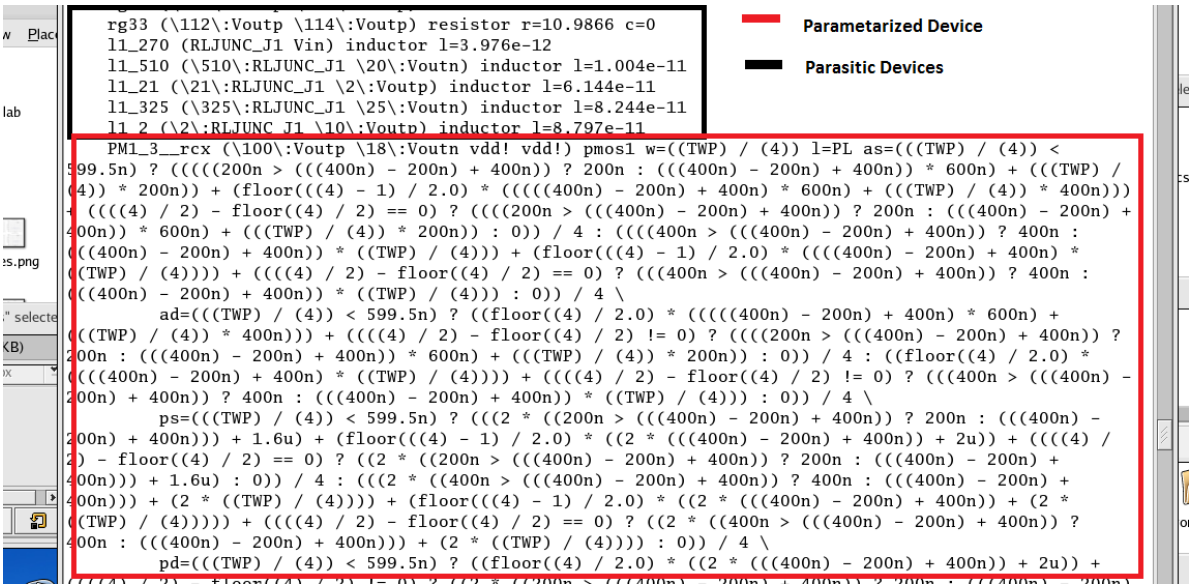


Figure 13: Netlist Example.

```
for(x 0 999
envOption(
        'analysisOrder  list("tran" "pss" "pnoise")
)
save( 'i "/V1/MINUS" )
temp( 27 )
run()

power = average((IT("/V1/MINUS") * VT("/vdd!")))
fprintf(powerPort "%e \n",power)

Voutp_Frequency = frequency(clip(VT("/Voutp") 5e-06 6e-06))
fprintf(freqPort "%e \n",Voutp_Frequency)

settling_time = settlingTime((freq VT("/Voutp") "falling" ?xName "time" ?mode "auto" ?
threshold 0.0) 0 nil 1 t 0.25 nil "time")
fprintf(stimePort "%e \n",settling_time)

jitter_vertical = eyeDiagram(v("/Voutp" ?result "tran-tran") 5e-06 6e-06 (1 / frequency(clip
(VT("/Voutp") 5e-06 6e-06))) ?advOptions 'vertical)
fprintf(jittervPort "%e \n",jitter_vertical)

jitter_horizontal = eyeDiagram(v("/Voutp" ?result "tran-tran") 5e-06 6e-06 (1 / frequency(clip
(VT("/Voutp") 5e-06 6e-06))) ?advOptions 'horizontal)
fprintf(jitterhPort "%e \n",jitter_horizontal)


drain(powerPort)
drain(freqPort)
drain(stimePort)
drain(jittervPort)
drain(jitterhPort)

);end for
```

Figure 14: Prepare the ocean file for LHS runs.

```
simulator( 'spectre )
design(  "/homes/omg0006/simulation/test_PLL_parametric/spectre/schematic/netlist/netlist")
resultsDir( "/homes/omg0006/simulation/test_PLL_parametric/spectre/schematic" )
modelFile(
    '("/apps/cds/gpdk/gpdk180_v3.3/models/spectre/gpdk.scs" "stat")
)
analysis('tran ?stop "6u"  ?errpreset "liberal"  )

powerPort = outfile("/homes/omg0006/Desktop/simruns/pll_power_cons.csv" "w")
freqPort = outfile("/homes/omg0006/Desktop/simruns/pll_freq_cons.csv" "w")
stimePort = outfile("/homes/omg0006/Desktop/simruns/pll_stime_cons.csv" "w")
jittervPort = outfile("/homes/omg0006/Desktop/simruns/pll_jitterv_cons.csv" "w")
jitterhPort = outfile("/homes/omg0006/Desktop/simruns/pll_jitterh_cons.csv" "w")

envOption( 'userCmdLineOption "-64" )

desVar(   "L" 180n      )
desVar(   "L3" 180n     )
desVar(   "L2" 180n     )
desVar(   "Wp2" 4u      )
desVar(   "Wn2" 2u      )
desVar(   "Wp3" 4u      )
desVar(   "Wn3" 2u      )
desVar(   "Wp_DIV_3" 4u )
desVar(   "Wp_DIV_2" 4u )
desVar(   "Wp_DIV_1" 4u )
desVar(   "Wp_DIV_0" 4u )
desVar(   "Wn_DIV_4" 2u )
desVar(   "Wn_DIV_3" 2u )
desVar(   "Wn_DIV_2" 2u )
desVar(   "Wn_DIV_1" 2u )
desVar(   "Wn_DIV_0" 2u )
desVar(   "Wp" 4u       )
desVar(   "Wn" 2u       )
desVar(   "Wp_VCO" 24u  )
desVar(   "Wn_VCO" 9u   )
desVar(   "Wn_s" 2u     )
desVar(   "Wp_s" 4u     )
desVar(   "Wn_b" 16u    )
desVar(   "Wp_b" 32u    )
```

Figure 15: Prepare the ocean file for LHS runs.

10 Circuit Optimization

Optimization algorithm is also conducted in MATLAB. Several algorithms can be used for optimization over the circuit netlist as well as over the metamodels. FoM that is needed to be found is entered in the Needed Value field. Accuracy field denotes the percentage that is acceptable for the found value. Max iterations field sets the upper limit for iterations. Two setup matrices: Parameter MAX and Parameter MIN define the limit for each parameter. They follow [Parameter 1, Parameter 2, ... , Parameter N] form. The results field provides output for result found, parameter values and time it took to complete. Simulated annealing and Tabu Search has been explored for PLL components optimization [GMK11]. Bee colony based optimization is presented also investigated [GMK12b]. Detailed discussions of several algorithm is presented in [Gar12].

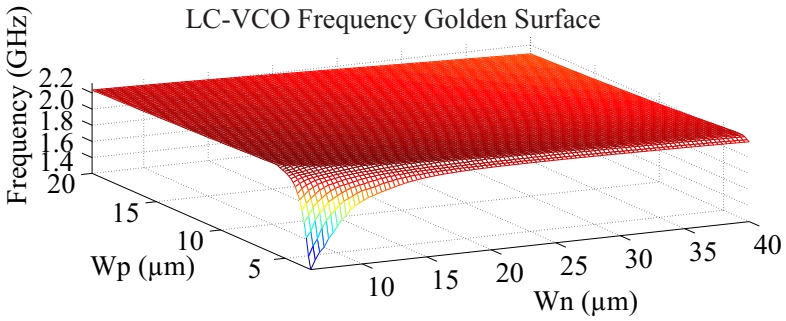
References

[BLB98] R. J. Baker, H. W. Li, and D. E. Boyce. *CMOS: Circuit Design, layout, and Simulation*. IEEE Press, 1998.

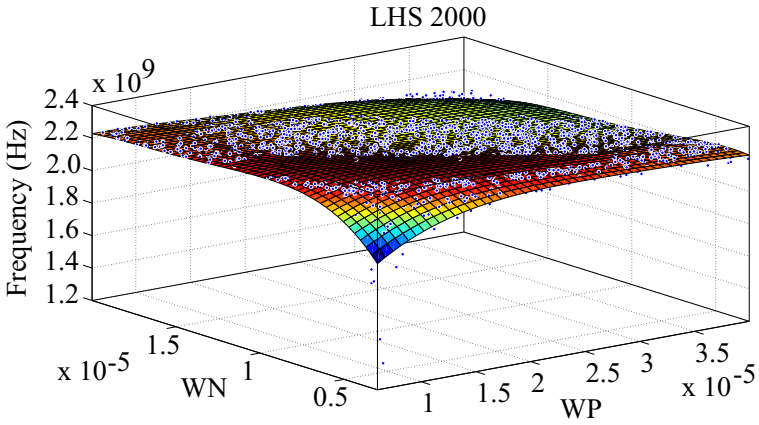
[Gar80] F. Gardner. Charge-Pump Phase-Lock Loops. *Communications, IEEE Transactions on [legacy, pre-1988]*, 28(11):1849–1858, November 1980.

[Gar12] O. Garitselov. *Metamodeling-Based Fast Optimization of Nanoscale AMS-SoCs*. PhD thesis, Department of Computer Science and Engineering, University of North Texas, Denton, 2012.

[GMK11] Oleg Garitselov, Saraju P. Mohanty, and Elias Kougianos. Fast Optimization of nano-CMOS Mixed-signal Circuits Through Accurate Metamodeling. In *Proceedings of the 12th International Symposium on Quality Electronic Design*, pages 405–410, 2011.



(a) Golden Surface



(b) LHS

Figure 16: Design space for 180nm CMOS LC-VCO.

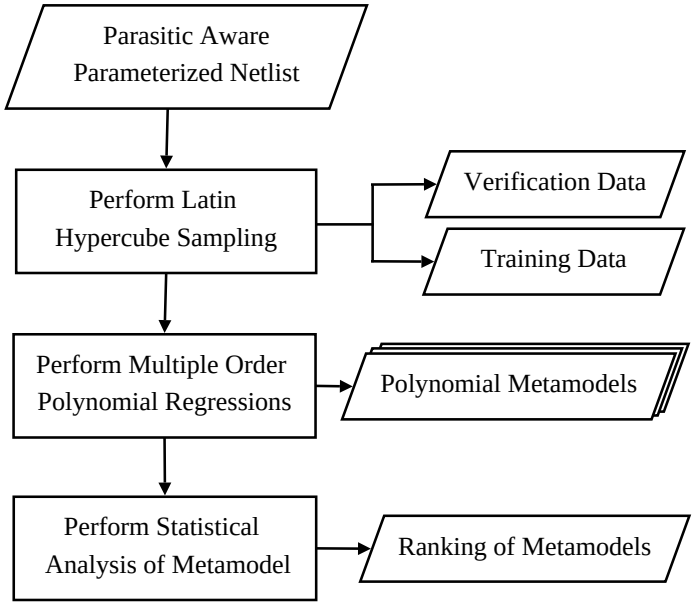


Figure 17: The proposed approach for metamodel generation.

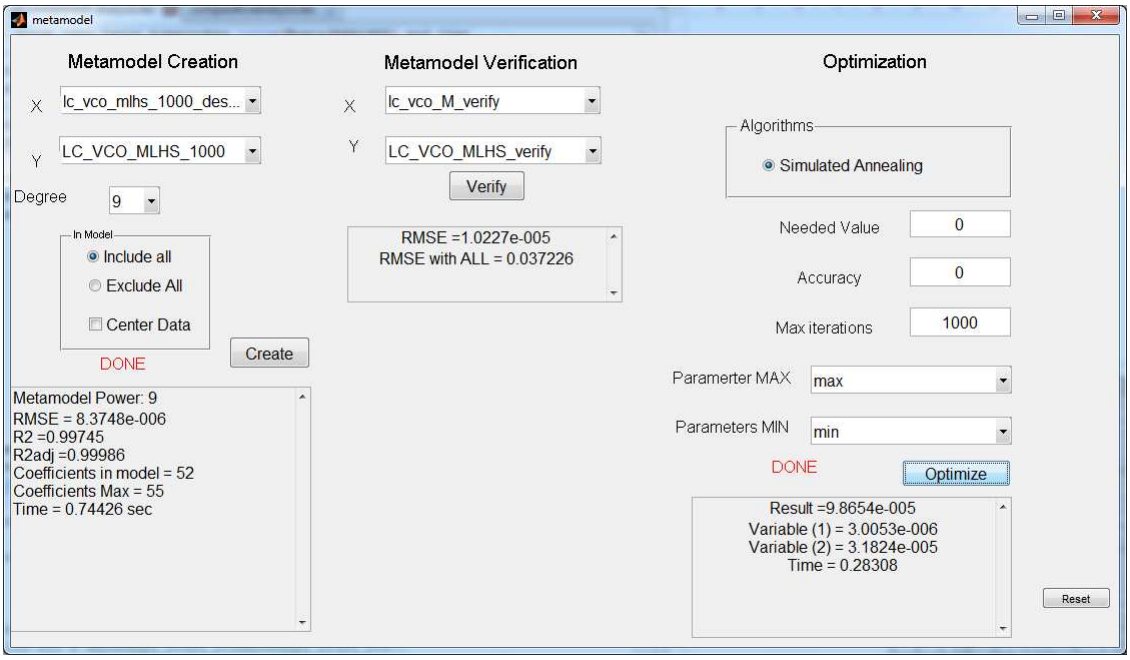


Figure 18: Screen capture of the metamodel toolbox.

[GMK12a] O. Garitselov, S. P. Mohanty, and E. Kougianos. A Comparative Study of Metamodels for Fast and Accurate Simulation of Nano-CMOS Circuits. *Semiconductor Manufacturing, IEEE Transactions on*, 25(1):26–36, Feb. 2012.

[GMK12b] Oleg Garitselov, Saraju P. Mohanty, and Elias Kougianos. Accurate Polynomial Metamodeling-Based Ultra-Fast Bee Colony Optimization of a Nano-CMOS Phase-Locked Loop. *J. Low Power Electronics*, 8(3):317–328, 2012.

[GMK12c] Oleg Garitselov, Saraju P. Mohanty, and Elias Kougianos. Fast-Accurate Non-Polynomial Metamodeling for Nano-CMOS PLL Design Optimization. In *Proceedings of the 25th International Conference on VLSI Design*, pages 316–321, 2012.

[Gup75] S.C. Gupta. Phase-Locked Loops. *Proceedings of the IEEE*, 63(2):291–306, February 1975.

[H. 05] H. Lee and T. Choi and S. Mohammadi1 and L. P. B. Katehi. An Extremely Low Power 2 GHz CMOS LC VCO for Wireless Communication Applications. In *Proc. Eur.*, pages 31–34. Microw. Conf., Oct 2005.

[Wan05a] David Tawei Wang. *Modern DRAM Memory Systems: Performance Analysis and a High Performance, Power-Constrained DRAM Scheduling Algorithm*. PhD thesis, University of Maryland, College Park, 2005.

[Wan05b] Zuoding Wang. An Analysis of Charge-Pump Phase-Locked Loops. *IEEE Transactions on Circuit and Systems-I: Fundamental Theory and Applications*, 52(10):2128–2138, October 2005.