METAMODELING-BASED FAST OPTIMIZATION OF NANOSCALE AMS-SOCS

Oleg Garitselov

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

May 2012

APPROVED:

Saraju P. Mohanty, Major Professor
Elias Kougianos, Co-Major Professor
Song Fu, Committee Member
Paul Tarau, Committee Member
Barrett Bryant, Chair of the Department of Computer Science and Engineering
Costas Tsatsoulis, Dean of the College of Engineering
James D. Meernik, Acting Dean of the Toulouse Graduate School

Garitselov, Oleg. Metamodeling-Based Fast Optimization of Nanoscale AMS-SoCs. Doctor of Philosophy (Computer Science and Engineering), May 2012, 94 pp., 19 tables, 44 illustrations, references, 62 titles.

Modern consumer electronic systems are mostly based on analog and digital circuits and are designed as analog/mixed-signal systems on chip (AMS-SoCs). The integration of analog and digital circuits on the same die makes the system cost effective. In AMS-SoCs, analog and mixed-signal portions have not traditionally received much attention due to their complexity.

As the fabrication technology advances, the simulation times for AMS-SoC circuits become more complex and take significant amounts of time. The time allocated for the circuit design and optimization creates a need to reduce the simulation time. The time constraints placed on designers are imposed by the ever-shortening time to market and non-recurrent cost of the chip. This dissertation proposes the use of a novel method, called metamodeling, and intelligent optimization algorithms to reduce the design time. Metamodel-based ultra-fast design flows are proposed and investigated. Metamodel creation is a one time process and relies on fast sampling through accurate parasitic-aware simulations. One of the targets of this dissertation is to minimize the sample size while retaining the accuracy of the model. In order to achieve this goal, different statistical sampling techniques are explored and applied to various AMS-SoC circuits. Also, different metamodel functions are explored for their accuracy and application to AMS-SoCs. Several different optimization algorithms are compared for global optimization accuracy and convergence. Three different AMS circuits, ring oscillator, inductor-capacitor voltage-controlled oscillator (LC-VCO) and phase locked loop (PLL) that are present in many AMS-SoC are used in this study for design flow application. Metamodels created in this dissertation provide accuracy with an error of less than 2% from the physical layout simulations. After optimal

sampling investigation, metamodel functions and optimization algorithms are ranked in terms of speed and accuracy. Experimental results show that the proposed design flow provides roughly 5,000x speedup over conventional design flows. Thus, this dissertation greatly advances the state-of-the-art in mixed-signal design and will assist towards making consumer electronics cheaper and affordable.

ACKNOWLEDGMENT

I would like to sincerely thank my major professor, Dr. Saraju P. Mohanty, for the encouragement, guidance and constant support throughout my studies at the University of North Texas (http://www.cse.unt.edu). I would also like to thank my co-major professor, Dr. Elias Kougianos, who has also supported my work by providing technical guidance and direction during my research.

I would also like to thank my Ph.D. committee members, Dr. Song Fu and Dr. Paul Tarau, for giving me this opportunity and providing technical guidance in writing this dissertation. Many thanks to my lab mates in NanoSystem Design Laboratory (NSDL, http://nsdl.cse.unt.edu): Karo, Garima, Geng and Asha. This dissertation would not be complete without their input.

I also wish to thank the staff of the Department of Computer Science and Engineering (http://www.cse.unt.edu) for being very helpful and caring in times of need. But most of all, I would like to thank my wife and son for their patience during the long and lonely nights they spent during my studies.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND MOTIVATION

Consumer electronic systems are typically mixed-signal systems with analog and digital components. The different mixed-signal system components perform different types of operations. Their design needs various skill sets and design methods. In this chapter, the typical structure of a mixed-signal system is discussed. The issues faced when the target fabrication technology for such systems is in nanoscale CMOS are presented. The problems that this dissertation addressed are presented. The highlights of the solutions proposed in this dissertation are also presented.

1.1. Mixed Signal Systems

Modern consumer electronic systems (e.g., mobile phones, BLURAY players, health monitoring systems) have profound impacts on society. Modern consumer electronic systems are typically designed as heterogeneous systems consisting of diverse analog and digital, and mixed-signal components and their interfaces on a single board or die. Typically, they are analog/mixed signal systems-on-chip (AMS-SoCs) or mixed-signal systems where analog and digital portions are integrated on the same die for cost-performance tradeoffs [38, 40]. There is also software (including both system software and application software) that facilitates the co-ordination of their operation and user interfaces. Because small integrated circuits (ICs, aka chips) are the main workhorses in consumer electronics, the efficient design of chips is one key driving factor in society due to their omnipresence, from kitchens to aircrafts. Present-day nano-CMOS ICs of gigascale complexity present the following design challenges: variability, leakage, power, thermal issues, reliability, and yield [37, 39]. These are aggravated due to the explosive growth of mobile appliances in which battery life dictates system performance. The situation is worsened for such highly complex chips, because the time-to-market has been reduced significantly. In this situation, methodologies for fast design space exploration are much more important than ever to timely produce error-free chips.

SoC systems are becoming more complex. The amount of analog circuits and mixed signal interface circuitry increases as the number of components shrink. Figure 1.1 shows the breakdown

of the mixed signal, digital and pure analog circuitry that can possibly be present in an SoC architecture. Even though the digital components take up more room due to the amount of transistors and logic complexity, the amount of different mixed signal circuits and the analog circuit present a major design and verification complexity. Numerous simulation tools for digital and analog circuits are used to aid in the design process, but as the technology progresses and scales down, the additional physical/electrical effects (parasitics) of circuit elements after the physical layout stage are making the calculations more complex [23]. The parasitics presented in the circuit have a very dramatic effect on the output. The design tools for analog and mixed signal circuits on the schematic design level cannot reach the accuracy of the physical design process since they do not account for parasitics of the circuit.

The design cycle of mixed-signal components is significantly long as accurate, circuit-level simulation is very CPU intensive. For example, the simulation time for a phase-locked loop (PLL) lock on a full parasitic netlist is of the order of many days to weeks. Thus, speedup of the design process can have a dramatic impact on time-to-market. This situation is further aggravated when such circuits are designed using nanoscale technology where the transistors are modeled using hundreds of parameters. It is also very difficult to accurately predict the performance of AMS-SoC component circuits in high frequency applications due to the many parasitic effects. To meet the desired design specifications, the original design is iteratively adjusted by attempting different values of the design variables. A large number of design variables results in an enormous amount of different possibilities for alternative design tradeoffs. Thus, in the context of efficient design of AMS-SoCs several questions arise:

i. In what way can fast layout of AMS-SoC components be generated?
ii. What are efficient ways to perform fast design optimization?
iii. What are efficient techniques to sample the design space?
iv. Is there an optimal metamodeling methodology for RF IC components?
v. What is an efficient RF IC design flow that can use metamodels effectively?
vi. What are the tradeoffs of design using metamodels versus actual circuits?
vii. What algorithms can be used for AMS-SoC design optimization?

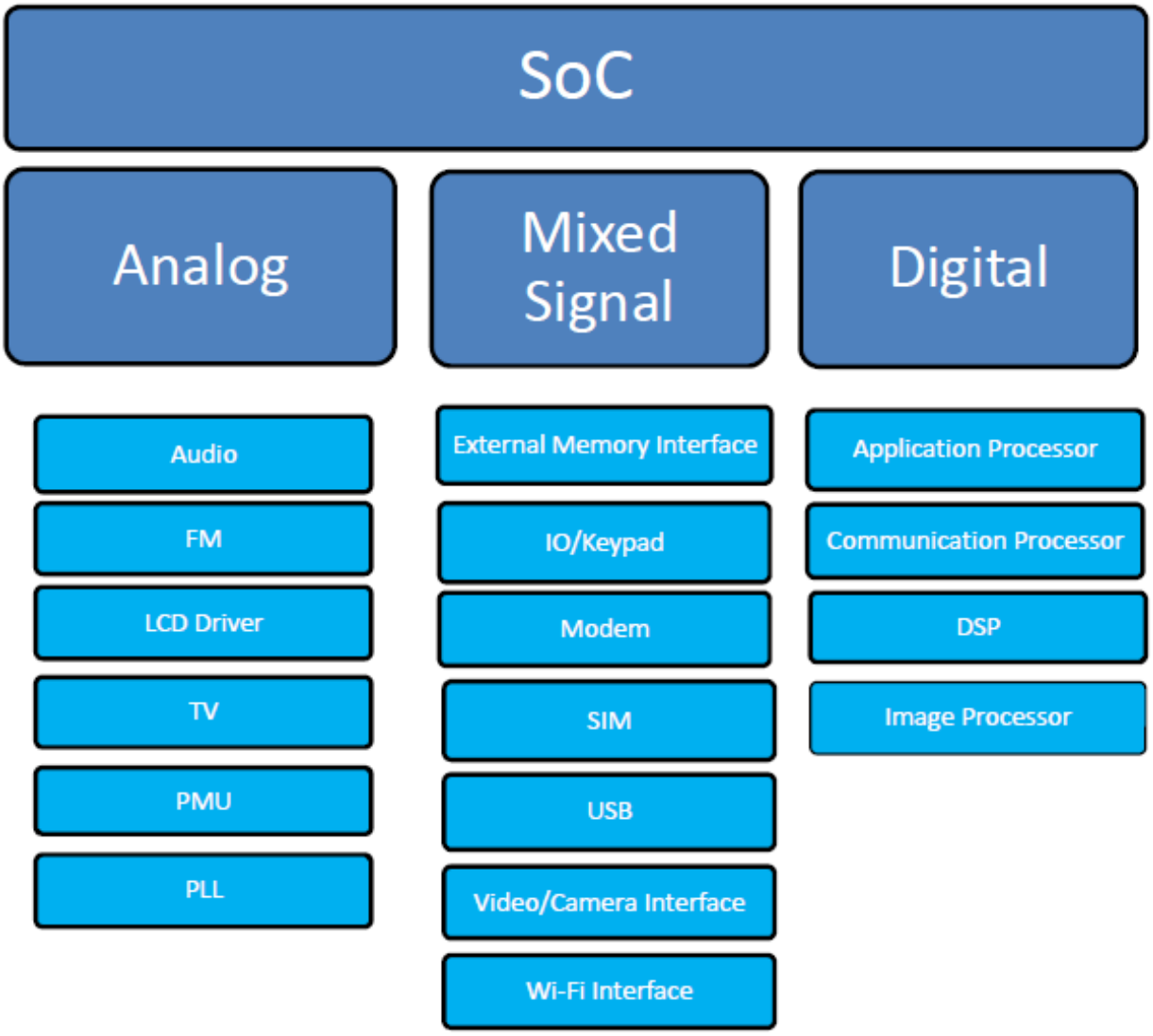


FIGURE 1.1. Breakdown of a typical SoC.

The design time constraints on the designers establish the need to decrease the simulation, design, and optimization times. These efforts will be silicon accurate if full-blown parasitics are taken into account. The full-blown parasitics include resistances (R), capacitances (C), self-inductances (L), and mutual inductances (K), which may be due to active devices or interconnects. Creating the physical design, particularly when nanoscale technology is used, involves very time consuming manual labor. The classical design flow needs much manual iterations involving physical layout and schematic adjustments. The number of manual iterations depend on the skill and qualifications of the design engineers.

Hence, designers usually encounter problems in the post layout stage of the design process that affect the design specifications and skew the output of the designed system. Because of that, the layout stage, which is time consuming process by itself, usually takes more than one iteration and in the worst case the designer has to go back to the drawing board. Therefore in many cases the final design becomes suboptimal with designers trying to fit their design into tight specifications and time to market deadlines.

The design process for analog circuits can take a very long time. Once the circuit design is complete it needs to be adjusted by parameterizing the design variables to produce the desired output. It is very difficult to predict the performance for analog circuits in high frequency applications due to many parasitic effects [46].

1.2. Nanoscale Mixed Signal Design Issues

Design engineers need to address the issues that arise when AMS-SoCs are targeted to be fabricated using nanoscale technology to ensure high yield and profit.

Figure 1.2 shows issues that arise in AMS-SoC component design space. As mobile devices are becoming more popular, power consumption and long battery life becomes a major target for optimization. In the case of digital circuits, low power design has already been addressed in various techniques, such as power gating, clock gating, adaptive voltage scaling (AVS), dynamic clock frequency and voltage scaling (DCVS), and static voltage scaling (SVS). Power consumption can be broken down into two different categories: dynamic and static. Dynamic power consumption occurs during the operational phase of transistors, when switching occurs. Static power occurs during the non-operational phase of transistors which includes leakage power due to capacitive current leaking through the gate of the transistor. Energy-efficient design for analog, mixed-signal, and heterogeneous systems still needs significant research.

As the technology is shrinking in size, static power consumption increases due to lower capacitance between the gate and channel of the transistor. While dynamic power consumption decreases, it takes less current to switch on the transistor. Digital approaches to power minimization do not work for analog components due to their high dependency on the supply voltage and transistor operation in the subthreshold region. Most analog circuits have to be custom designed

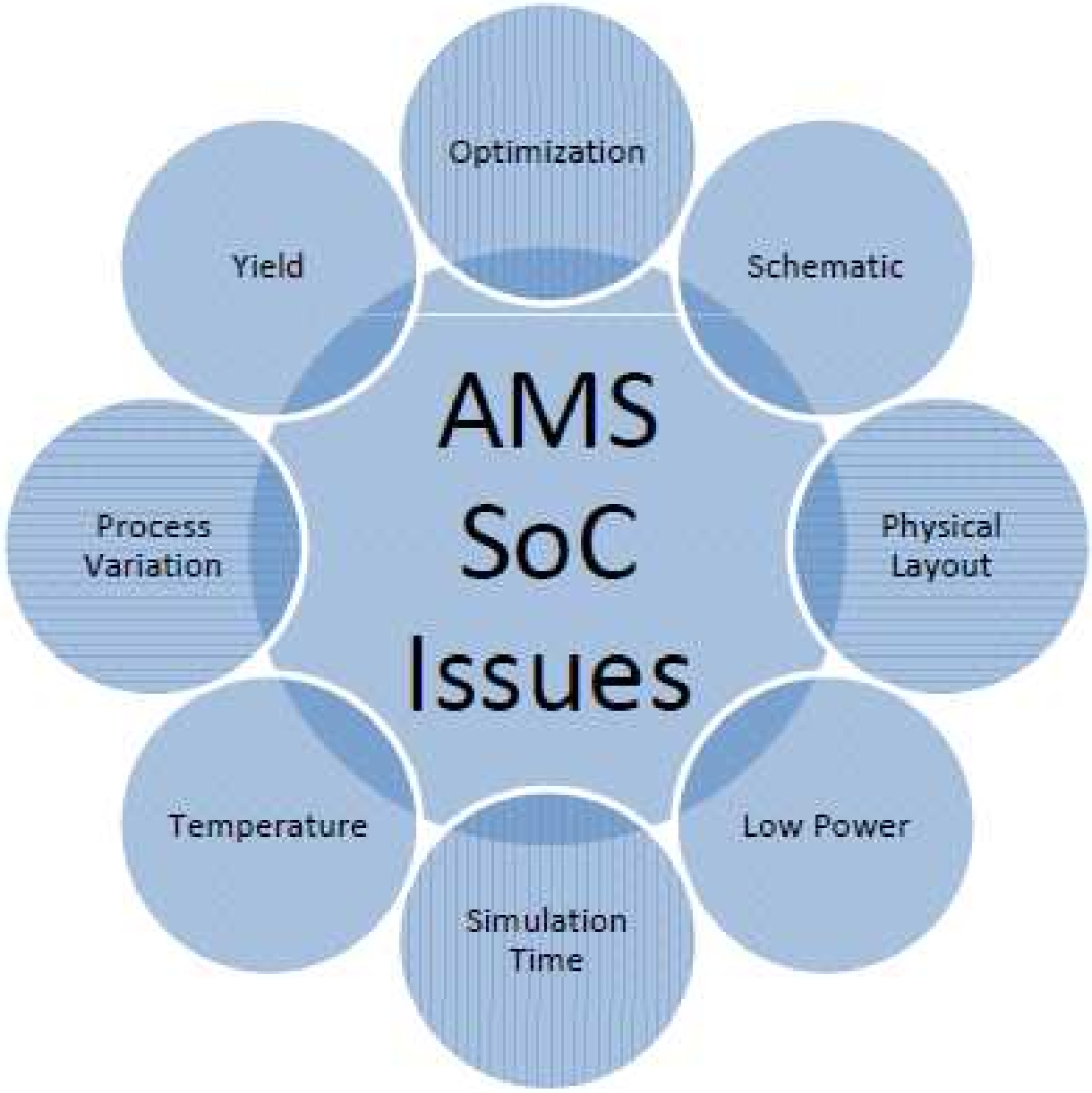


FIGURE 1.2. Typical AMS-SoC design and fabrication issues.

for every application. There is very little intellectual property (IP) reuse in analog mixed signal circuits, unlike in digital circuits where standard cell libraries are used for a given process. The standby power dissipation and leakage is an important issue for battery driven electronic systems that stay in waiting for most of the time. For example, smart phones continuously waiting for an incoming call. It has been observed that the gate-oxide leakage can have severe impact on the oscillating frequency of nanoscale PLLs [33].

The development of high-speed data communication, information processing, and multimedia interaction devices has increased the need for inexpensive high frequency clock generation. When systems are operating at a higher clock frequency, the clock skew associated with clock distribution is a critical problem. In addition, data synchronization between chips and on-chip clocks with higher frequencies than the external clock are usually required in many system designs. Hence, a high performance phase locked loop (PLL) with capabilities of low clock skew, low phase noise, and low clock jitter is a very crucial unit in a high-speed system design [47].

Accurate optimization for AMS circuits has been studied by many researches. Since the design process takes a very long time, especially the optimization and layout creation, the objective to minimize the development time and increase accuracy is a very common research topic. Analog circuits are very difficult to predict and it is best to use simulation software such as Simulation Program with Integrated Circuit Emphasis (SPICE) for optimization. SPICE is very computationally expensive to be used for the actual circuit, since it uses Berkeley short-channel insulated-gate field-effect transistor (IGFET) models (BSIM4/5) of transistors to create the netlist of small design units, which are then used for assembly of subsystems. The SPICE model of the circuit is thus very large and for complex circuits can take days if not weeks to compute.

Since it is very expensive to manufacture the components, process variation analysis has to be conducted on the physical layout. This analysis creates statistical data that can predict the yield of the manufacturing process. Monte Carlo (MC) analysis and design corner analysis are two most common methods that are performed. To perform MC analysis enough data need to be collected to generate an accurate statistical distribution. Usually, this involves a large number of simulations obtained by varying the design parameters randomly. Design corner analysis involves a smaller amount of simulations; it generates data on the min/max corners of each parameter, creating a rough idea of circuit behavior. The design corner analysis method is a lot faster to perform if the designer does not need to know the behavior of the distribution of a circuit metric.

Operating temperature as well as heat spots can create large problems in the design. Even though the design can work to the specifications in normal temperatures, the threshold voltage of the transistor is decreased as operation temperature increases. This affects the switching speed of

the transistor and creates irregularities that affect mixed signal and analog design characteristics more than digital designs. In general, with an increase in temperature, the effect on the nanoscale AMS-SoC is triply negative as follows:

i. It increases the device leakage.
ii. It reduces the active current strength.
iii. It increases the propagation delay, thus making the device slower.

If a positive feedback loop is established between leakage and localized temperature, a complete thermal breakdown can occur. An increase in temperature will lead to an increase in leakage, and an increase in leakage can contribute to a higher temperature.

All of the above problems hinder the designers' ability to produce the design within specifications and within the allotted design time. The manufacturing price of the circuit is very high and it is essential to produce the design to be manufactured with the lowest tolerance for process error to reduce manufacturing cost. The uncontrollable process variability in device characteristics represents major challenges to scaling and integration for present AMS-SoC circuits. This in turn demands revolutionary changes in the way in which current and future integrated circuits and systems are designed. Strong links must be established between circuit design, system design and fundamental device technology to allow circuits and systems to accommodate this increasing variability.

1.3. The Problems Addressed in this Dissertation

The target for this dissertation is to simplify the optimization phase of the process and conduct it with the least amount of simulation time as possible. The previous sections briefly described the issues that can arise during the design process. Since simulation times for complex circuits are excessive, it is not feasible to conduct an exhaustive search to find the optimized output. The numerical methods are usually not very accurate since in most cases the formulas do not account for the parasitics of the circuit, which can only be calculated after the initial design is complete. Figure 1.3 shows the scope of the problems addressed in this dissertation.

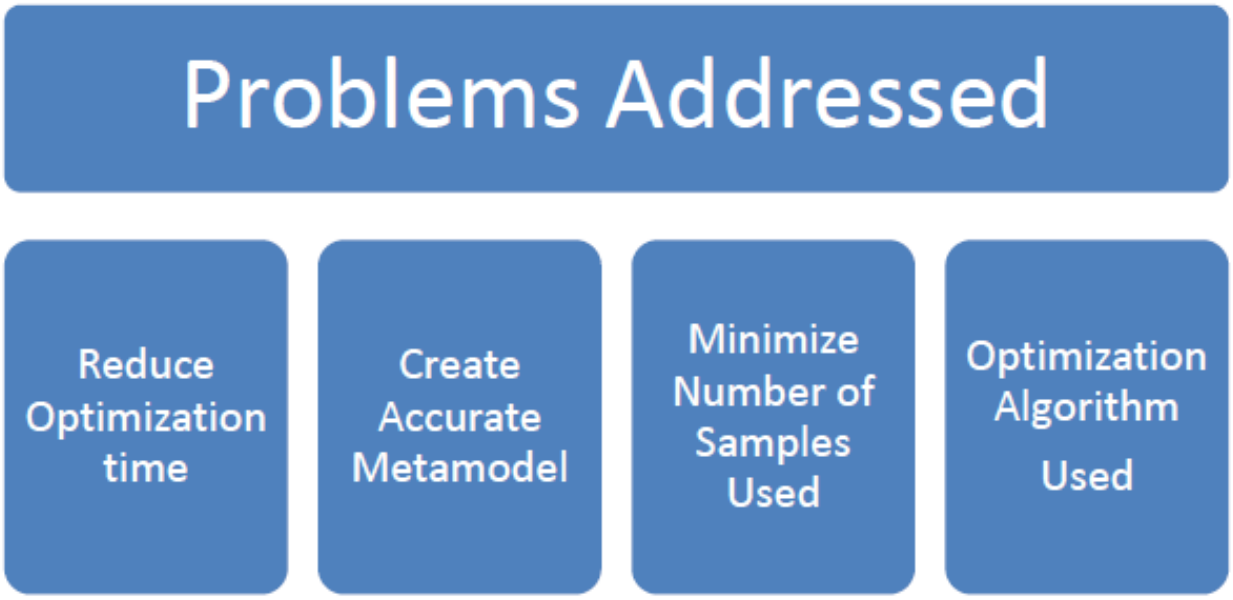


FIGURE 1.3. Problems addressed in this dissertation.

The proposed approach is compared to the simulation design approach. Further solutions are shown in Figure 1.4 and both are discussed further in this section. The pure simulation approach is usually followed when a designer has a lot of time to spend for running simulations to optimize the circuit. Unfortunately the timelines for current designs are very short. Hence, the need to find accurate and less time consuming design flows led us to the metamodeling approach that is used widely in other technical fields. The proposed metamodeling design flow has been used successfully in different analog circuits, and proposes only two iterations for the physical layout. The first iteration is the starting physical design and the later is the final design after the optimization phase.

Metamodels are presented in this dissertation. A metamodel is essentially a predictive mathematical formula for a given figure of merit (FoM) such as power, frequency, jitter, leakage, phase noise, etc. [17]. Each circuit can obviously have more than one metamodel if the optimization step is multiobjective which requires a metamodel for each needed FoM to be optimized. Using mathematical functions in the optimization step speeds up the process since each iteration of the calculations does not require computing the circuit's characteristics for every iteration, which is a slow process, or recreating the physical design. Creating a metamodel is a very crucial step, since

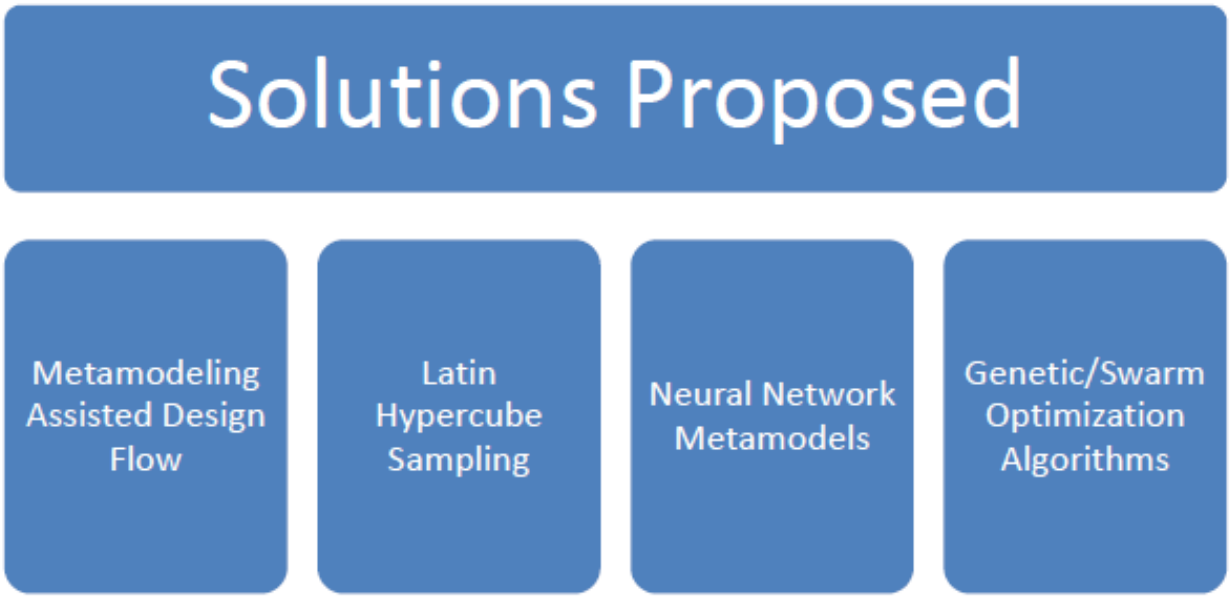


FIGURE 1.4. Solutions proposed to the problems addressed.

the manufacturing price of the circuit is very high and it is essential to produce the most accurate metamodel for that design to be manufactured with the lowest tolerance for error. Metamodeling is not a novel approach, and is used frequently in other production and software fields which have time consuming or very expensive manufacturing process.

Different types of metamodels are considered in this work. Polynomial functions of various degrees and constitution (full and stepwise) and also various neural networks are considered. Exhaustively searching for better metamodels, from varying the order of the polynomial to creating multiple neural models with different configurations showed that feed forward neural models provide better overall results for a phase-locked loop (PLL) system. The PLL is used for analysis to show that metamodels can be used on a complex circuit. Two different specifications are used: 2.7 GHz - MMDS and 2.5 GHz - WiiMAX and it is shown that the given design flow can be used on a circuit that does not fall within the specifications and is flexible and robust.

To create a metamodel the design space has to be sampled. This work also conducts s comparative study on sampling the design space since it is unknown how many samples are needed

to create a metamodel and what type of sampling to use to generate the most accurate one. After conducting sampling on different analog mixed signal circuits, it can be concluded that Latin Hypercube Sampling (LHS) shows better results than other sampling techniques.

This dissertation also demonstrates how metamodeling can be used to adjust the initial physical design circuit to the target figure of merit. To show that the proposed approach works, it is tested on the following two circuits: a 45 nm Ring Oscillator and a 180 nm LC-VCO. By using two different technologies, the study proves that metamodeling is technology independent. To account for parasitics, the physical layouts are initially created for both circuits and a parasitic extracted netlist is then used for the analysis.

A very large number of design variables can produce an enormous amount of different possibilities for the design. As it is a very expensive process to manufacture the circuit, it is essential to create the closest possible result to generate a circuit that can be manufactured only once with the lowest tolerance of error. Using fast and accurate algorithms for optimization is essential for the optimization process. Once the metamodels are created, swarm intelligent, memetic and simulated annealing algorithm optimization techniques were applied onto the metamodel and the actual circuits netlist to find the optimal solution for the given problem as a comparison. The optimization is conducted on various metamodels and later the results are compared.

1.4. Organization of this Dissertation

This dissertation is organized as follows. The dissertation layout is represented in Figure 1.5. The state-of-the-art containing previous and related research is presented in Chapter 2. Chapter 3 includes a detailed discussion on the design of the case study circuit, a PLL which contains both analog and digital components and hence is a classic example of a mixed-signal system. The proposed metamodeling design flow is proposed and compared to a regular design flow in Chapter 4. Design space sampling techniques are investigated in Chapter 5. Chapter 6 discusses polynomial and neural network metamodels. Various optimization algorithms that are used for mixed signal circuit optimization are explained in Chapter 7. Conclusion and future research can be found in Chapter 8.

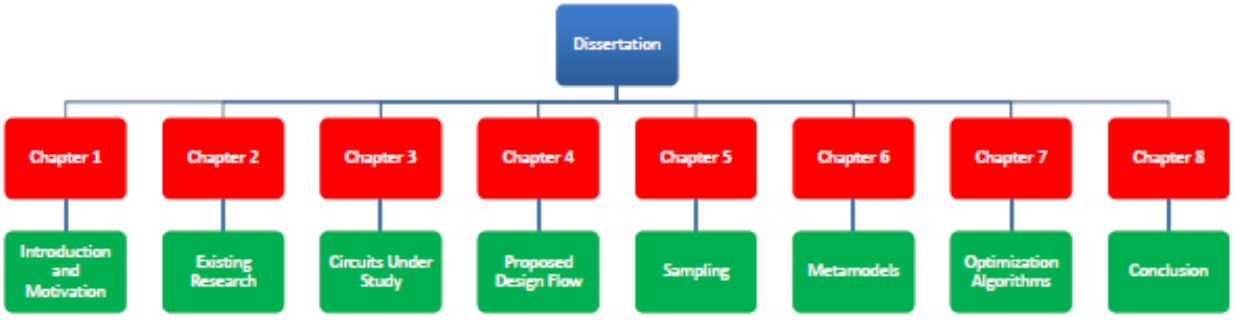


FIGURE 1.5. Dissertation organization.

1.5. List of Acronyms and Symbols Used in this Dissertation

This section lists all acronyms and symbols used throughout this dissertation. Most of the acronyms are typically used in the standard literature. However, their definitions are described here for completeness.

Acronym	Definition
ABC	Artificial bee colony
AMS	Analog/mixed signal
BCA	Bee colony algorithm
FoM	Figure of merit
GA	Genetic algorithm
L_{eff}	Effective gate length of a transistor
L_n	Gate length of an NMOS transistor
L_p	Gate length of a PMOS transistor
LHS	Latin hypercube sampling
MC	Monte Carlo
MLHS	Middle Latin hypercube sampling
MMDS	Multichannel multipoint distribution service
MSE	Mean square error
NMOS	N-type metal oxide semiconductor transistor
PLL	Phase locked loop

PMOS	P-type metal oxide semiconductor transistor
RMSE	Root mean square error
RO	Ring oscillator
SA	Simulated annealing
SoC	System-on-chip
SSE	Sum of square error
T_{ox_n}	Oxide thickness of the NMOS transistor
T_{ox_p}	Oxide thickness of the PMOS transistor
V_{DD}	Voltage supply of the circuit
V_{CS}	Stored voltage of the memory cell
V_{th_n}	Threshold voltage of the NMOS transistor
V_{th_p}	Threshold voltage of the PMOS transistor
W_n	Gate width of an NMOS transistor
W_p	Gate width of a PMOS transistor
WiMAX	Worldwide interoperability for microwave access

CHAPTER 2

RELATED PRIOR RESEARCH

Research has been undertaken around the globe to invent methods for fast design exploration of mixed-signal systems with minimal effort. Different research works attack different phases of the design cycle of mixed-signal components. In this chapter selected related research is discussed in order to provide a broad perspective. The fast optimization track that is targeted in this dissertation is shown in Figure 2.1. It can be divided into four different tracks: sampling, actual netlist optimization, macromodeling, and metamodeling.

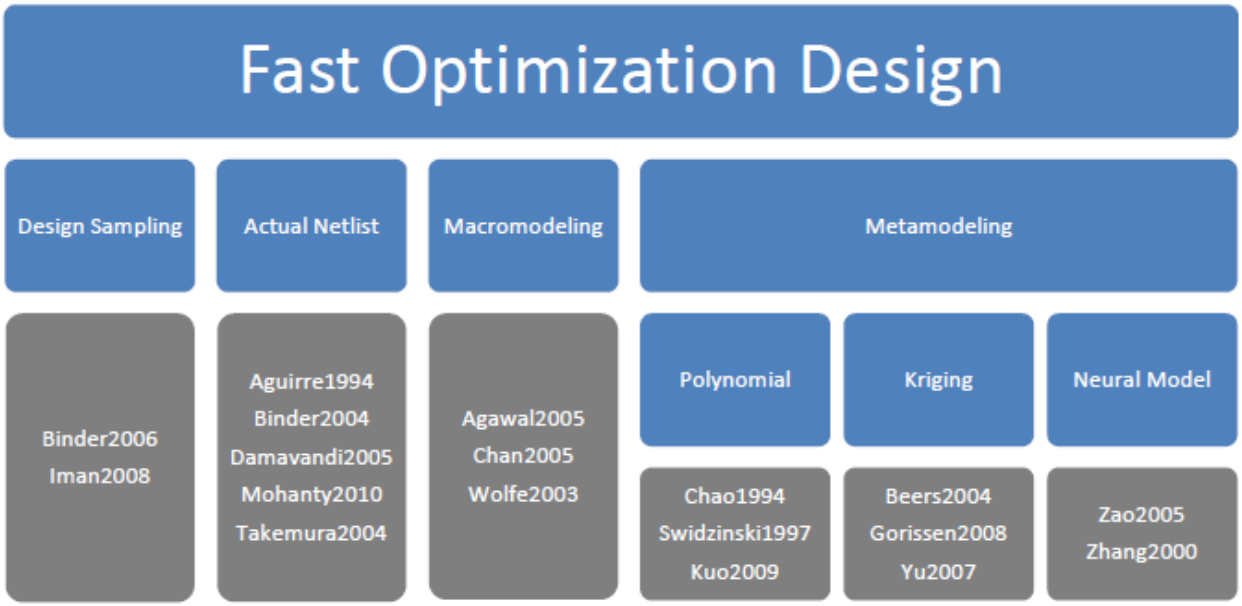


FIGURE 2.1. Fast optimization design tracks.

2.1. Design Sampling

Design sampling is the process of accurately capturing design points in the multidimensional design space of the mixed-signal components. The challenge is to accurately capture the sample points. At the same time, effectively traversing the design space is the key. Sampling techniques can be divided into three different categories: random, uniform and Design of Experiments (DOE) [19].

There are different kinds of uniform sampling techniques and many other derivatives. Two common uniform sampling techniques are Latin hypercube sampling (LHS) and middle Latin hypercube sampling (MLHS). The different variations that derive from these two techniques include the following: orthogonal array-based Latin hypercube design, symmetric Latin hypercube design, orthogonal column LHS, and optimal Latin hypercube design. In [29] the Latin hypercube sampling (LHS) and its deviations are described in full detail.

Monte Carlo or random sampling is a technique which samples the data for each variable, by selecting random data points for each variable in the domain. Monte Carlo is a very common method used in process variation analysis in circuit design. Monte Carlo sampling methods have been extensively described in [4].

Design of experiments (DOE) is a technique used when a large number of variables are involved in the design process. DOE is a statistical method based on the analysis of variance and applied for controlled experiments. DOE is explained thoroughly in [14]. DOE is usually used to minimize the amount of samples needed to predict the design's behavior.

2.2. Optimization of Circuit Netlist

Many researchers try to use complex optimization algorithms to reach the optimal value efficiently. But the optimization iteration uses at least one or more circuit simulations that are conducted on the circuit schematic or layout parasitic netlist. These results usually do not guarantee that the optimal value was reached within given allotted optimization time.

The design flow for optimization of a 90 nm VCO is presented in [41]. The paper takes power, performance, process, parasitic, voltage and temperature variations into account. A dual optimization technique is used in which the frequency is maximized and power consumption is lowered. After the logical schematic meets the specifications, the parasitic netlist is extracted from the physical circuit layout. The netlist is then edited to include the needed parameters instead of hard coded layout values. Monte Carlo process variation analysis is then carried out on the created netlist. A 10% process variation is used to differentiate and mark the worst case from the mean. Transient analysis is run on the netlist that includes parasitics; power consumption is measured for each transistor in the circuit. The most power hungry transistors are then identified. Higher V_{th} is

applied to those transistors to reduce power consumption, followed by the optimization stage. The authors use a conjugated gradient optimization algorithm that is executed until the given criteria are met. The algorithm takes into account the process variation from the data that was extracted using Monte Carlo analysis. The parasitic values, from the base line design, are usually used when the values are not changed dramatically. However, the parasitic effects for the final stage can have different effect on the circuit in the final layout. It is not shown how much error is introduced by using this approach for the given circuit. This research work successfully decreased power consumption by 16.4% with only 10% degradation in center frequency in the effect of worst case variations. This is a significant enhancement to the performance from the baseline design.

Faster corner-based optimization of a 90 nm analog-to-digital converter (ADC) is proposed in [22].

A hybrid evolutionary programming (EP) optimization algorithm is applied to a diplexer circuit in [9]. The paper targets premature convergence of EP algorithms, where the algorithm terminates on a suboptimal solution. The proposed algorithm is tested on mathematical functions and the diplexer circuit. In [51], a hierarchical particle swarm optimization algorithm is proposed for automatic sizing of low-power analog circuits. In [11], the bee colony optimization algorithm is investigated for the transient performance analysis of a CMOS circuit.

A Genetic Algorithm (GA) is used on Winner Line-up Amplifier (WLA) and Winner Take All (WTA) circuits in [49]. The paper parameterizes the circuit netlist and uses GA algorithm to optimize circuits to specification. It shows that analog circuits can be optimized using high level optimization algorithms successfully. The optimization conducted is on pre-parasitic extraction phase of the design process. Another new genetic algorithm is proposed in [15] for floor planning of gigascale complexity that simultaneously minimizes the total wire length and silicon area.

In [5], a comparative analysis of multiple optimization methods is provided. The optimization is used for the calibration of TCAD simulators and is not used for circuits. A heuristic Tabu search optimization algorithm is proposed in [2] and compared with simulated annealing for an operational amplifier. The optimization is not performed on an actual SPICE netlist and analytical formulas are used for characterizing the optimization.

Previous research shows that optimization algorithms can be used successfully for circuit optimization. The optimization algorithms still use a significant amount of time to reach the near optimal values. For very large circuits the optimization time is still not acceptable, therefore, other researchers try to find different ways to reduce simulation time. The metamodel assisted minimal manual iteration flow presented in the current dissertation fills the need.

2.3. Macromodeling and Optimization on Macromodels

Many common approaches use macromodeling to reduce the complexity of the system under study Macromodeling or circuit simplification [55, 1, 12, 3] uses simplified circuits that mirror the circuit's behavior to reduce its simulation time. It is usually conducted in the same simulator tool, i.e. SPICE. This approach is mainly used to reduce simulation time of the design block so as to reduce simulation time of the whole system. Macromodels are not parameter sensitive and therefore cannot be used for optimization purposes. Macromodels are usually used in conjunction with metamodels that translate the size parameters of the circuit to the corresponding macromodels. Hence, the macromodeling approach might include higher error, which is the sum of macromodel and metamodel errors.

An example of pure macromodeling circuit optimization of an 180 nm differential Op Amp is presented in [7]. The macromodels were created for the circuit and parasitics were extracted from the layout of the circuit. Extensive setup and circuit knowledge was required for the proposed design flow. The authors showed that the proposed design flow was able to account for parasitic changes during each optimization iteration.

As a paradigm shift, this dissertation focuses on metamodeling. In the metamodeling approach, the underlying system is completely decoupled from the simulator and the resulting metamodel (i.e., mathematical model of the circuit netlist) is more general, flexible and easier to simulate and optimize than macromodels. In summary, a metamodel has the following salient features:

i. It is a mathematical representation of the circuit output in terms of the design parameters.
ii. It is a prediction equation or formulas not, circuit netlists.
iii. Metamodels can be used in a variety of tools and are language independent.

2.4. Metamodeling Techniques

There have been many attempts to simplify the design stage to optimize the design back to specifications after the initial layout. Modeling approaches for the circuit as a system (metamodeling) are also investigated. To name a few: regression [8], neural networks [62], and Kriging [52, 24]. All these methods use mathematical and/or statistical analysis which can use math-intensive tools like MATLAB. The modeling creates a formula or set of formulas to predict the output of the system with given parameters. The parameters usually used are device sizing, but it can be any other parameter of the circuit such as, temperature or even an input signal. Each predictive formula follows the $f^{'}(x) = f(x) + e(x)$ format, where function $f(x)$ is actual function and $e(x)$ is the error function. The accuracy of the model can is affected by the error function which in turn needs to be minimized. The formula(e) later can be used to predict the system's behavior while adjusting the design parameters. Metamodeling is widely used in other engineering fields, especially when the design simulation is either expensive or time consuming.

Hierarchical statistical analysis is a faster way than the Monte Carlo analysis and is used for process variation [48], [34]. It usually follows a polynomial regression based approach which creates equations from the response surfaces. In this approach, complicated circuits are divided into hierarchies; parameterized netlists are generated for higher level hierarchies. The models are then created from the netlists forming prediction equations for each hierarchy and for each needed output. It is a complicated process to set up and each prediction equation carries the error from one hierarchy to another.

Neural networks for RF and microwave design are described in [60]. It is a thorough survey of neural network applications in modeling microstrip lines via CPW discontinuities, spiral inductors, printed antennas, FET, and VLSI interconnects. In [56], the use of neural networks in the automatic synthesis of op-amps is explored. In [58], a feed-forward dynamic neural network model is introduced for amplifier and mixer circuits directly from input-output large-signal measurements. In [57], the creation of neural networks for the estimation of the output of operational amplifiers is presented.

In [24], the authors show that Kriging shows better results than ordinary regression. This is because the interpolation for predictive points depends on global and local weights around a needed point. The weights depend on the approximation of sample points to the point of interest, and need to be recalculated each time the interpolation point changes with each iteration of the optimization algorithm. This makes Kriging more mathematically intensive than regression but also more accurate. Kriging was applied in [59] for a ring and LC oscillator and two stage Op-Amp behavior analysis and optimization. The analysis is applied to the yield aware Pareto front method.

In [42] the authors conduct a comparison between multiple metamodeling techniques. Radial basis neural network functions, support vector machines regression, polynomial regression, and Kriging functions, are compared for fitness for multi-objective evolutionary algorithm problems. This study compares metamodels for function's robustness, scalability, efficiency and suitability. The metamodels are utilized on eight problems that are described in [28]. Kriging metamodels show worst results due to slow calculation speeds. For low dimension problems, the polynomial regression and Kriging show good results. On the other hand, radial neural models and support vector regression perform best for higher dimension problems. Although the problems used are not on the circuits themselves, this work shows that common polynomial regression models, that are most commonly used in circuit modeling, do not perform as well as other more complicated metamodels.

2.5. Contributions of this Dissertation

The contributions of this dissertation to the state-of-art are multiple.

i. This dissertation proposes technology independent metamodeling sampling techniques using two different analog circuits (RO and LC-VCO) which are implemented in different technologies.

ii. Five distinct random and uniform metamodeling sampling techniques are introduced and are applied to nano-CMOS circuits. They include Monte Carlo (MC), Latin Hypercube Sampling (LHS), Middle Latin Hypercube Sampling (MLHS), and Design of Experiments (DOE). The use of these sampling techniques in metamodeling is demonstrated for a 45 nm CMOS ring

oscillator and an 180 nm CMOS LC-VCO. The ring oscillator is characterized for frequency, power and jitter, while the LC-VCO is characterized for frequency, power and phase noise. The full RCLK (resistance, capacitance, and self and mutual inductance) parasitic extractions are performed and compared to the schematics for both oscillators. The metamodels are generated on the parasitic netlist.

iii. This dissertation also proposes a metamodeling-based design flow for fast and accurate optimization of nano-CMOS complex mixed-signal circuits. The different phases of the design flow are identified and undertaken for future research, such as statistical sampling for metamodeling, mixed-signal optimization algorithms, and optimization of metamodels.

iv. For more optimal design the metamodel has to be accurate. Different types of metamodels are explored for their accuracy: polynomial, stepwise, radial and feed-forward neural networks.

v. As a step toward optimization, four distinct optimization algorithms are discussed. They include simulated annealing, bee colony and memetic algorithms and are applied to nano-CMOS technology. The use of optimization techniques with and without metamodeling is compared. The best optimization technique is identified from the above list.

vi. A 45 nm ring oscillator, 180 nm LC-VC0 and 180 nm Phase Locked Loop are designed and characterized, including the layout, and are used as case studies. The full RLCK parasitic extraction is performed and compared to the schematic of the circuits. Metamodels are generated on the parasitic extracted netlist from the initial physical layout.

vii. A Memetic algorithm is used and is shown to perform considerably better than the swarm intelligence optimization algorithm when applied to AMS circuits.

viii. It is shown that metamodeling based ultra-fast design flow is roughly 90% faster compared to customary design approach.

CHAPTER 3

MIXED-SIGNAL CASE STUDY CIRCUITS: PHASE LOCKED LOOP (PLL)

In this chapter a typical mixed-signal system, the phase-locked loop is presented. 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 us to use PLL as a case study system. The PLL consisting of analog and digital components makes a fitting example of a mixed-signal system. 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.

3.1. PLL System Overview

The Phase locked loop (PLL) system is presented in Figure 3.1. It 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 [54].

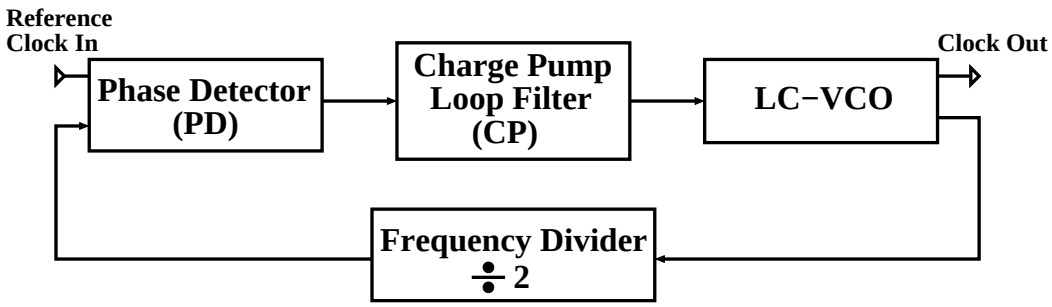


FIGURE 3.1. Block diagram of a PLL.

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 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 [25]. 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. Figure 3.2 shows the behavior of a PLL circuit.

In this dissertation, the PLL is designed and simulated in many ways. 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

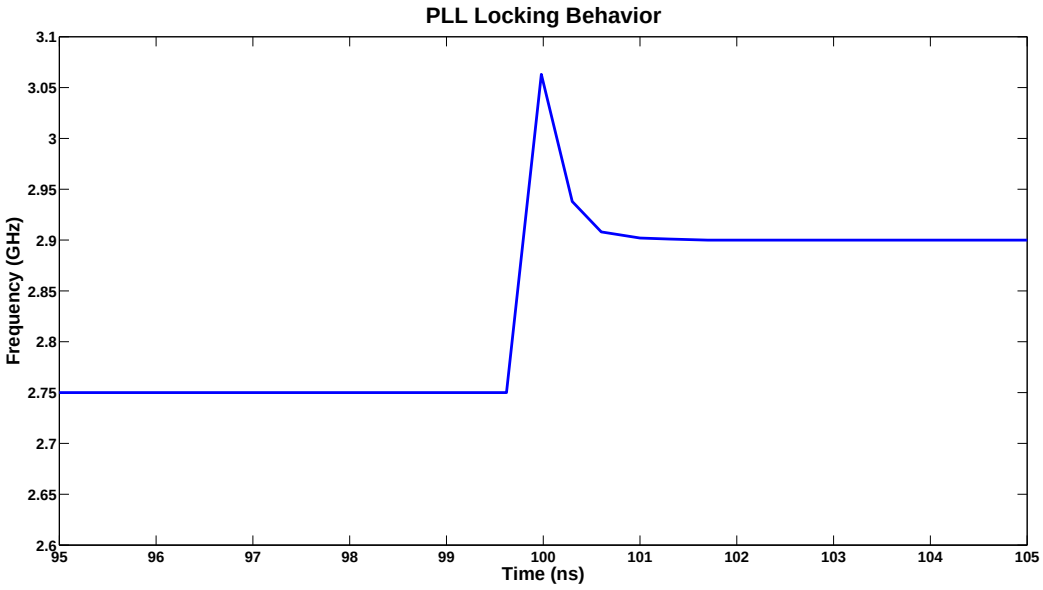


FIGURE 3.2. PLL response to change in frequency.

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 is calculated as follows:

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

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:

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

3.2. 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 [53]. 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 Figure 3.3. The gain of the phase detector is calculated using the following expression:

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

3.3. Charge Pump

The charge pump transistor level schematic is shown in figure 3.4. 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

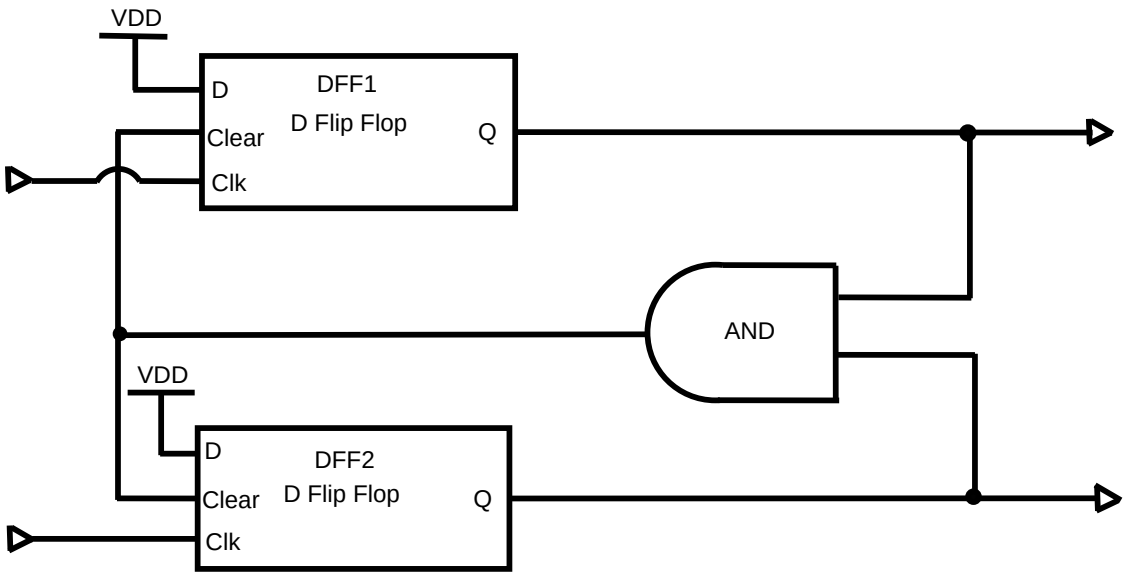


FIGURE 3.3. Phase detector.

for controlling the locked oscillators [16]. The gain for the charge pump is calculated using the following expression:

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

3.4. Loop Filter

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, as shown in figure 3.5. 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 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,

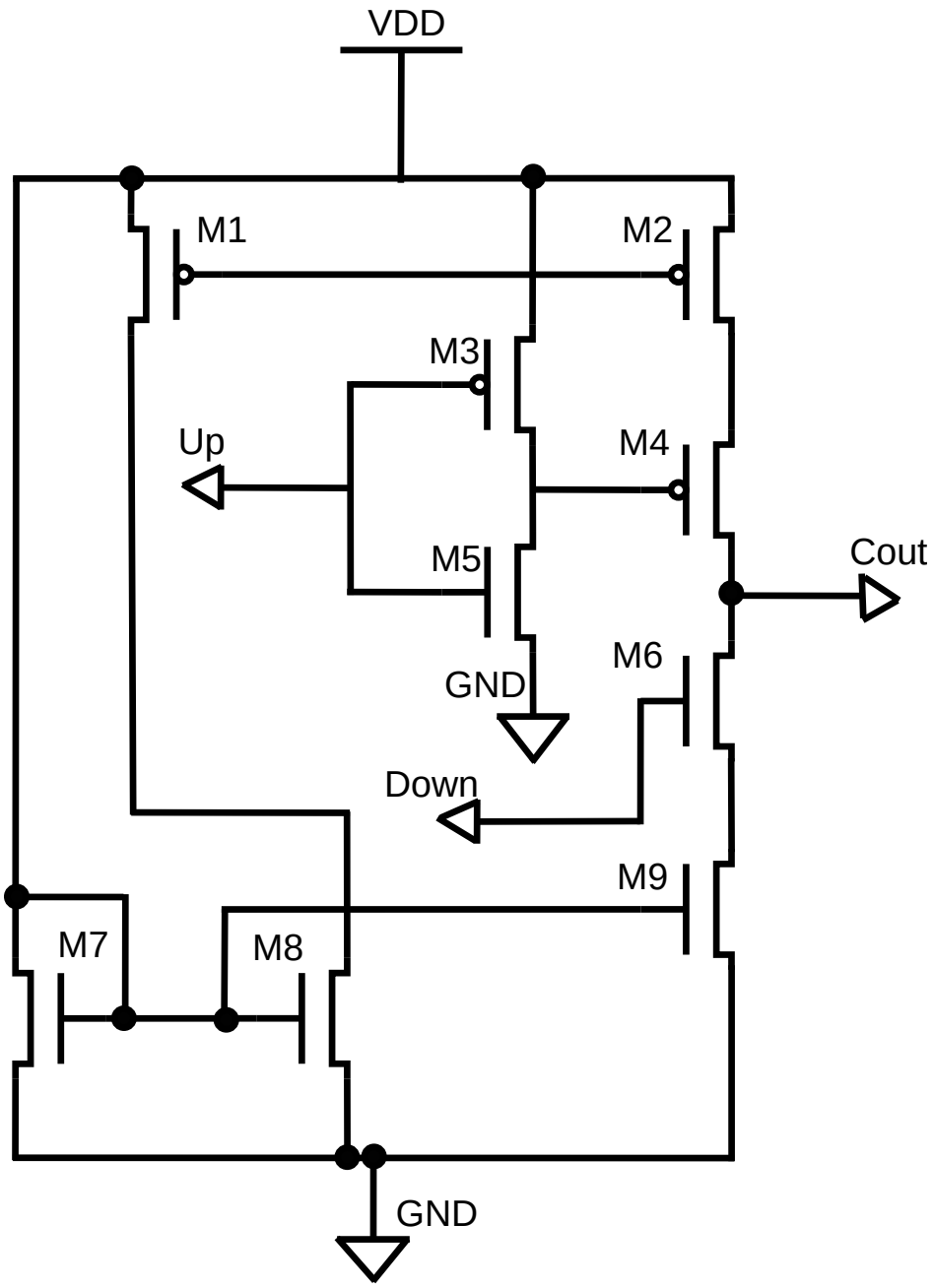


FIGURE 3.4. Schematic diagram of the charge pump.

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

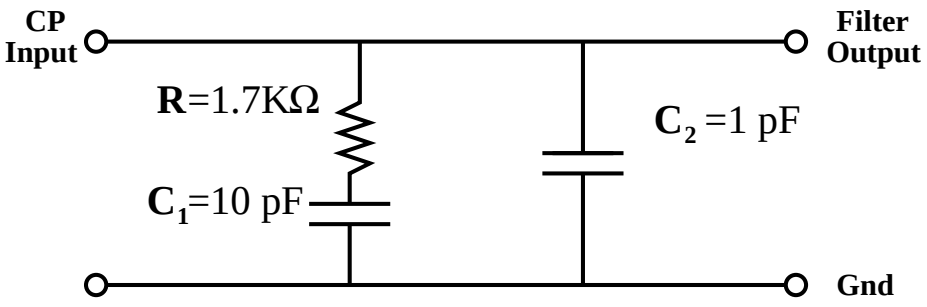


FIGURE 3.5. Schematic of loop filter

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.

3.5. Oscillator Circuits

There are many different architectures for PLL system. The PLL systems differ mainly based on their oscillator base. The oscillator base determines the frequency capture range and cleanliness of the output signal. Hence, the oscillator is the main part that is responsible for the output signal. Two different configurations of the oscillators are presented in the following sub-sections: LC-VCO and ring oscillator. The PLL oscillator needs to be selected carefully for each design for its characteristics, target applications, and area overhead.

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. 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.

In this dissertation, a conventional complementary NMOS and PMOS cross-coupled LC-VCO circuit is used and shown in Figure 3.6. 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:

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

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

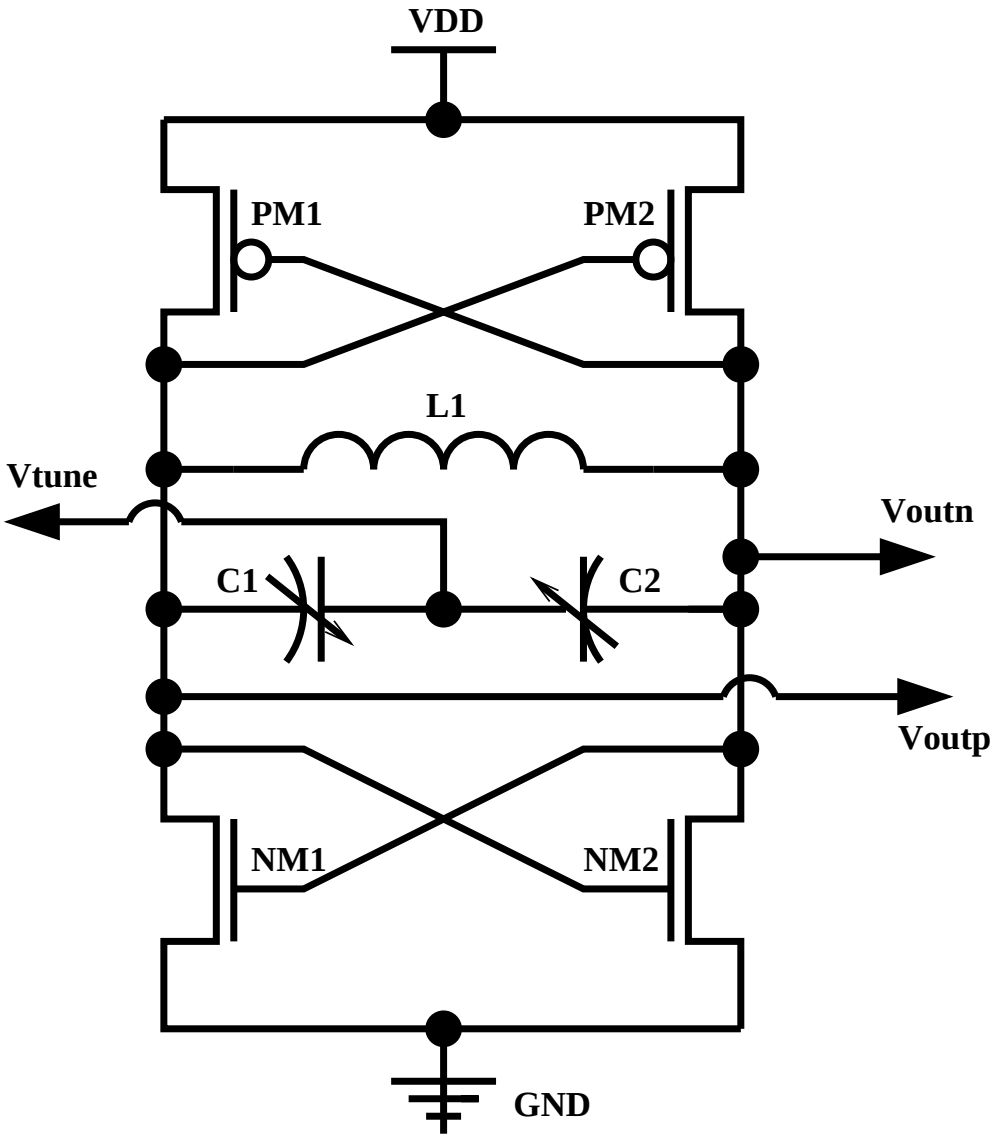


FIGURE 3.6. Schematic of the LC-VCO circuit.

GHz. Figure 3.7 shows the target frequency in the middle of the VCO's tuning range for schematic output, but as shown later, the frequency dropped dramatically because of the parasitic effects.

The phase noise is shown in Figure 3.8 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.

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 Figure 3.9. The netlist parasitic-aware netlist is obtained from this layout. The parasitic effects of the circuit were taken into consideration

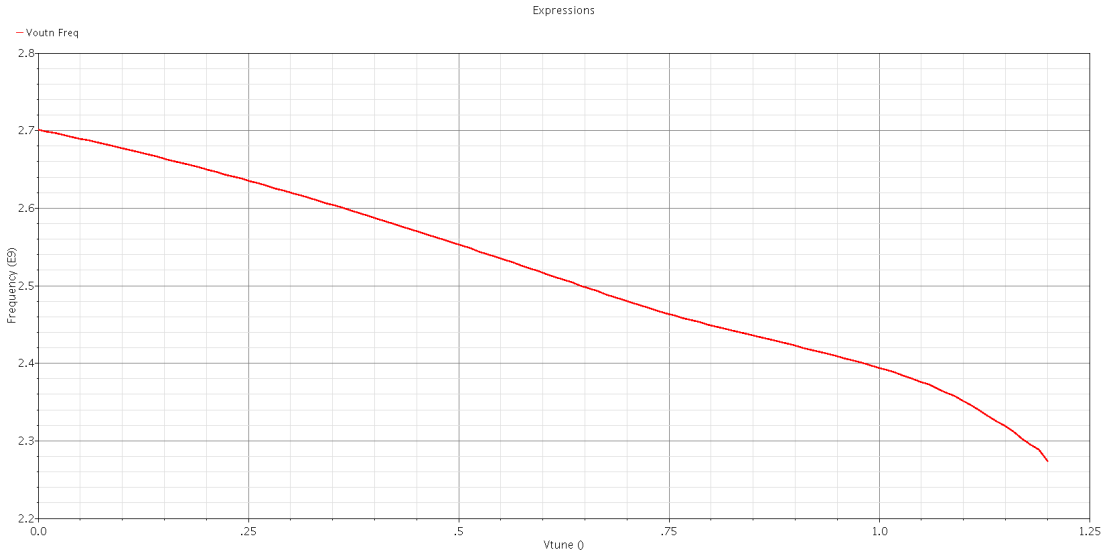


FIGURE 3.7. Measured frequency tuning characteristic of the LC-VCO.

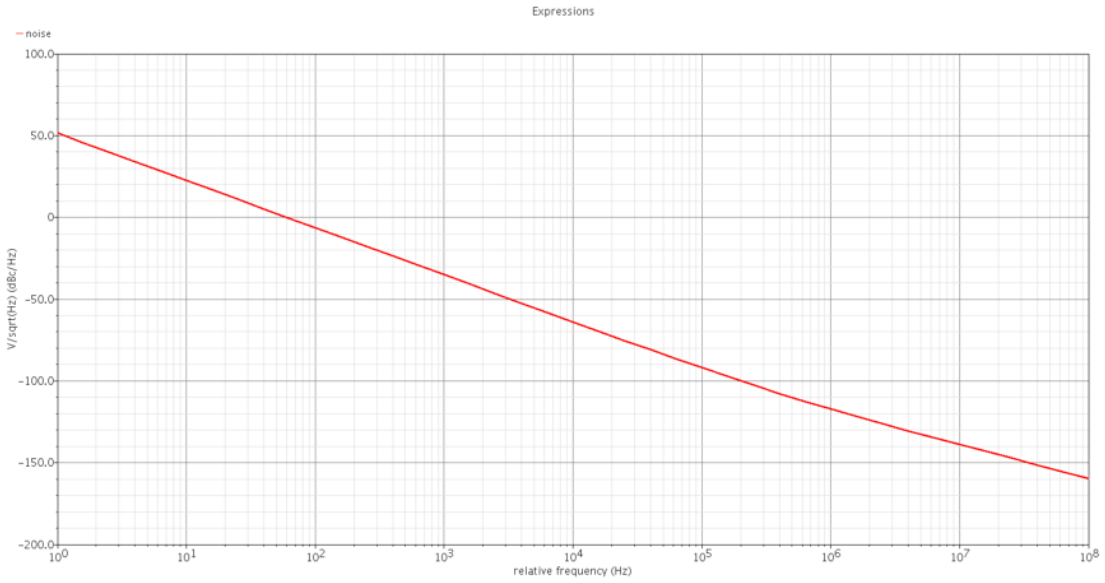


FIGURE 3.8. Measured phase noise of LC-VCO output at 2.5 GHz.

and the layout was made symmetrical. The symmetry of the layout provides up-conversion and the even-order distortion in the differential output waveform [26]. 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.

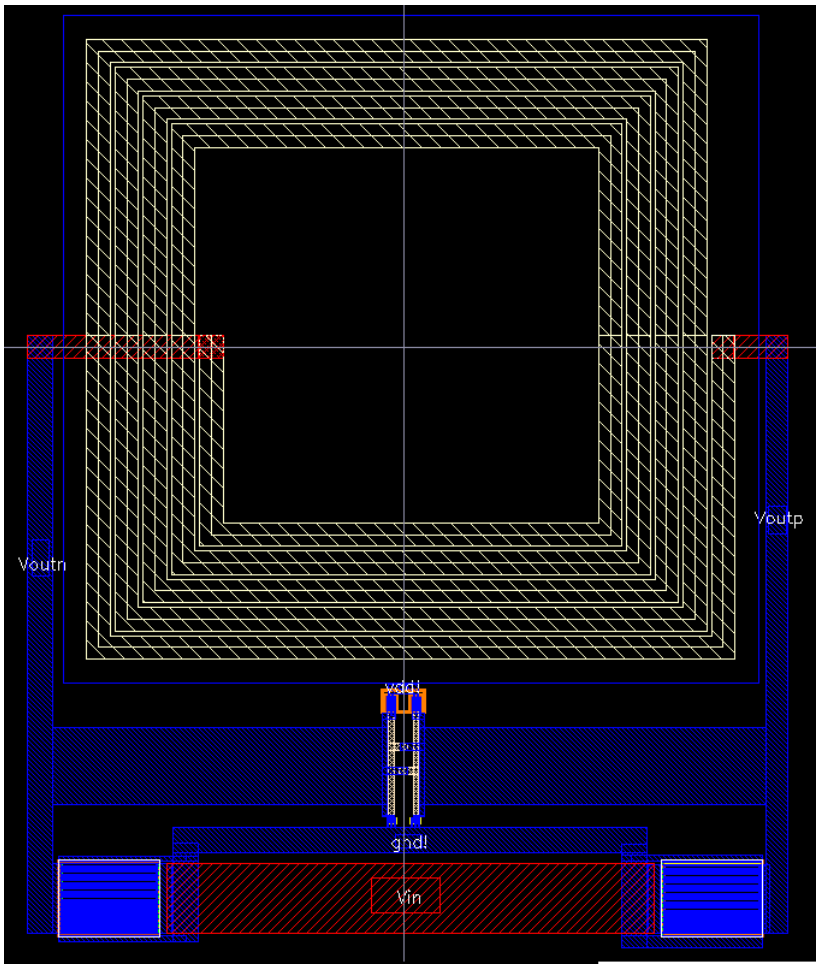


FIGURE 3.9. LC-VCO physical design for 180nm CMOS technology.

Ring Oscillator

A Ring Oscillator (RO) consists of an odd amount of inverters with a feedback loop as shown in Fig. 3.10. The feedback loop creates oscillations that are derived from the propagation

delay of each inverter. ROs are useful in die and new technology testing and are commonly used to find the delay times of logic gates. They are also the main element in temperature sensors.

Figure 3.11 shows the schematic diagram of a three inverter ring oscillator. For a given technology node, the designer can adjust the widths of the NMOS and PMOS to obtain the desired frequency. The design space is spanned by the two widths of the equal inverters.

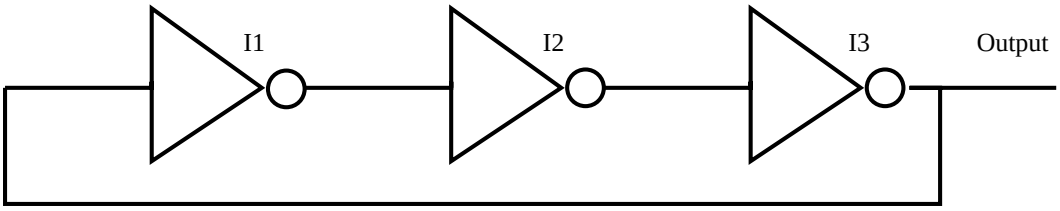


FIGURE 3.10. High-level schematic of the ring oscillator.

Assuming that all inverters have the same size transistors, their fall and rise times should be the same. Then the frequency of oscillation of the ring oscillator is calculated using the following expression [30]:

$$f = \left(\frac{1}{2Nt_p} \right), \tag{15}$$

where N is the number of inverters, which is odd, and t_p is the propagation delay of each inverter.

The design variables chosen for this design are: width of NMOS $W_n = 4L = 120$ nm and width of PMOS $W_p = 8L = 240$ nm at a nominal operating voltage $V_{dd} = 1$ V and minimal technology length of $L = 45$ nm. The ambient temperature of 27 degrees Celsius is assumed to be constant for all future simulations since the temperature difference can affect the output dramatically. Temperature analysis for self-heating effects are not taken into account in this dissertation.

It is difficult to estimate parasitic effects without actually performing the physical design. The parasitics in analog nano-CMOS circuits have a very dramatic effect on performance. Using the 45nm Process Design Kit (PDK) the ring oscillator physical design is performed. The layout is presented in Figure 3.12. The full SPICE netlist is generated from RCLK parasitic extraction from this layout. The physical design which involves tedious labor-intensive work by using *metamodeling design flow will only need to be conducted twice*, i.e., once for the initial design and one final time after obtaining the optimized data from the metamodel.

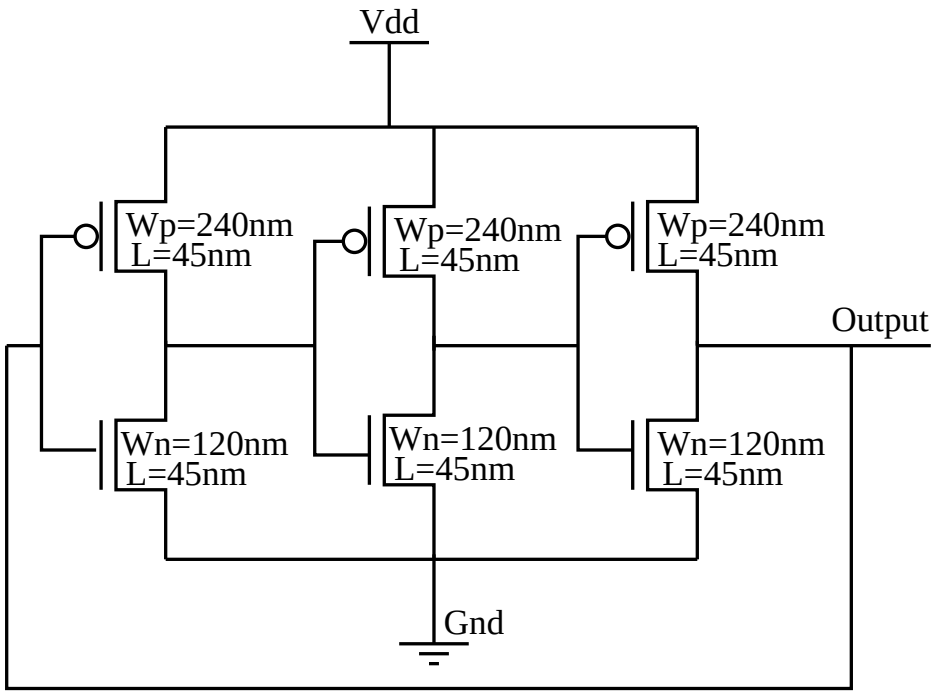


FIGURE 3.11. Transistor-level schematic of the ring oscillator with baseline sizes shown for a 45nm CMOS technology.

A dramatic decrease in frequency is observed in simulations using the parasitics from this layout versus the regular schematic simulations. For this simple circuit the simulation run time has increased by a factor of 3 due to the presence of parasitics. Contemporary complex circuit with thousands of transistors can have the simulation time in days, if not weeks, depending on the complexity of the circuit. Comparison of the number of components between the regular schematic and the parasitic netlist is shown in Table 3.1.

TABLE 3.1. Number of elements in the example circuits.

	Simulation	Transistors	Capacitors	Resistors	Inductors	Total
Ring Oscillator	Without parasitics	6	0	0	0	6
	With parasitics	6	82	19	0	107
LC-VCO	Without parasitics	4	2	0	1	7
	With parasitics	4	108	600	14	726

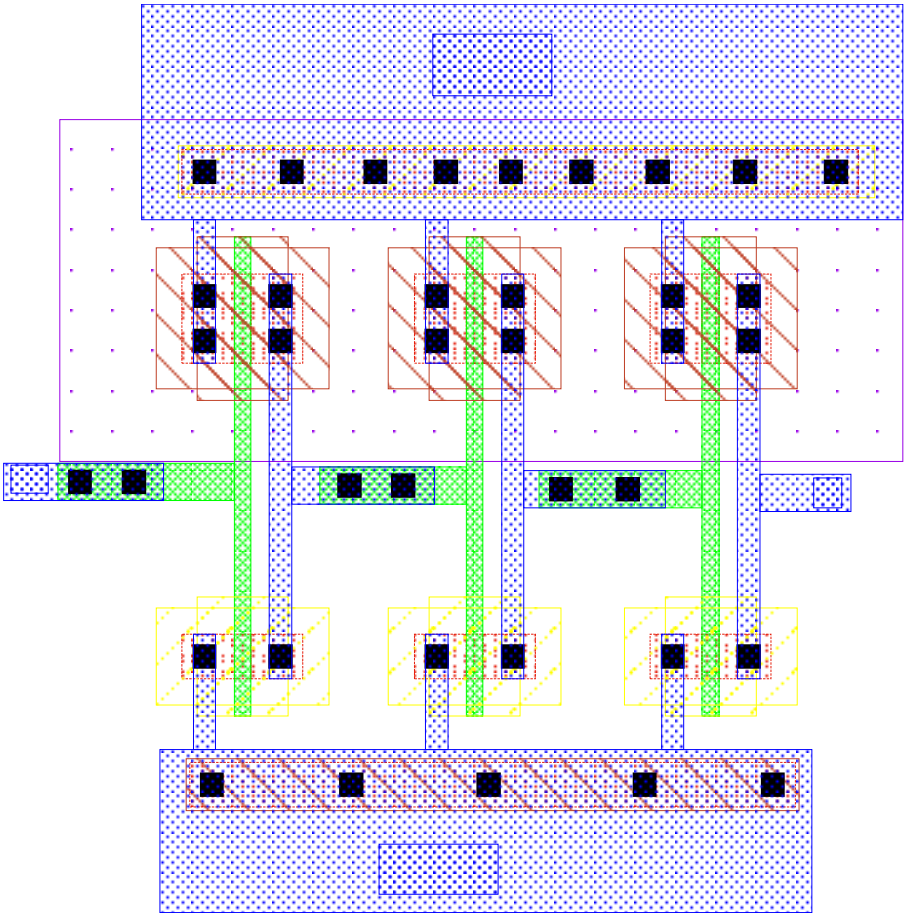


FIGURE 3.12. Ring oscillator layout for a $45nm$ CMOS technology.

There was no drastic effect observed from adjusting the widths of the CMOS components to $W_n = 360$ nm and $W_p = 720$ nm in the physical layout on the output frequency. On the other hand, the change between the schematic and parasitic outputs is of the order of 40%. It is observed that the total power consumption has not been altered, and only changed by merely 1%. Table 3.2 shows the comparison between the simulation results with and without parasitics. From these results, it can be concluded that the extraction of parasitics is necessary to calculate the desired output such as the frequency for this circuit.

The eye diagram for jitter effect of physical layout is shown in Figure 3.13. From the diagram, it can be seen that the jitter effect is negligible for a 100 ns period. The effect on jitter was minimal even when full parasitics are taken into account. Hence this dissertation will address frequency, as it is the most affected FoM.

TABLE 3.2. Simulation comparison for widths of NMOS=120nm and PMOS=240nm for the ring oscillator.

Extraction	Power	Frequency
Schematic	27.17μW	16.21 GHz
Parasitic	26.96μW	9.88 GHz

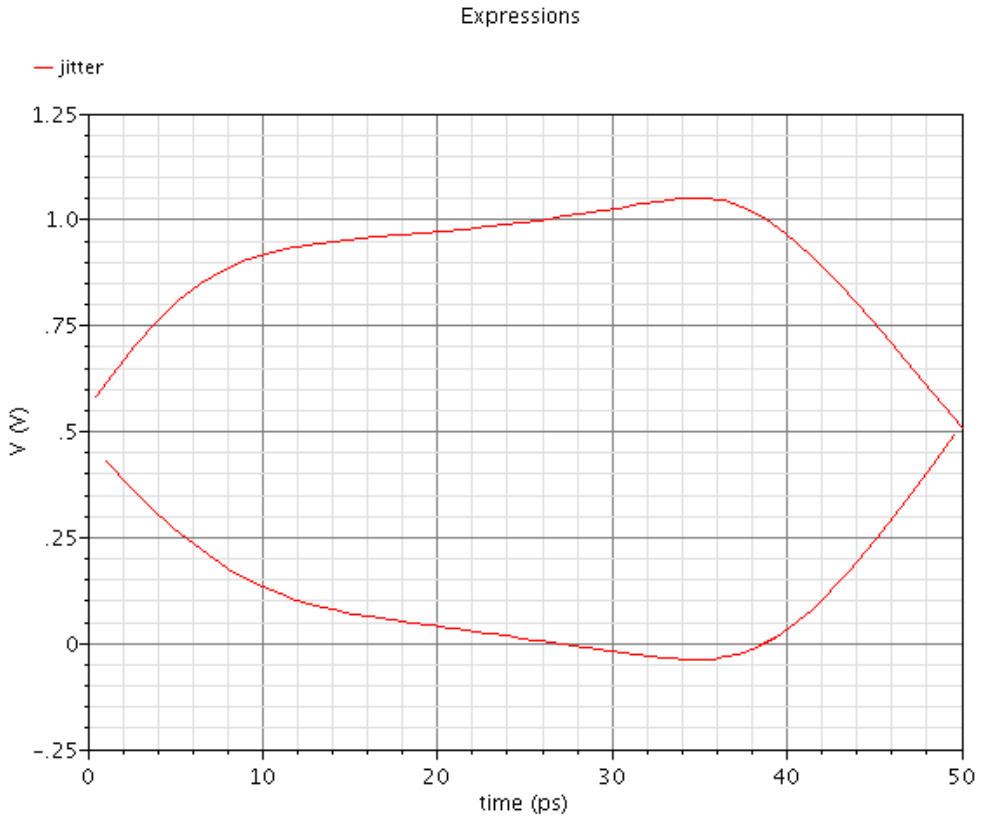


FIGURE 3.13. Eye diagram of parasitic netlist. From the eye diagram it is evident that the jitter is negligible.

3.6. Characterization of the Overall PLL

The PLL response to an ideal reference VCO's input voltage is shown in Figure 3.14 for the purpose of comparison. This input voltage is obtained from a Verilog-A based ideal simulation of an oscillator.

Figure 3.15 shows the phase noise diagram of the created PLL system. A phase noise of -162 dBc/Hz is observed at 1MHz offset which uses the LC-VCO with -117 dBc/Hz at 1MHz

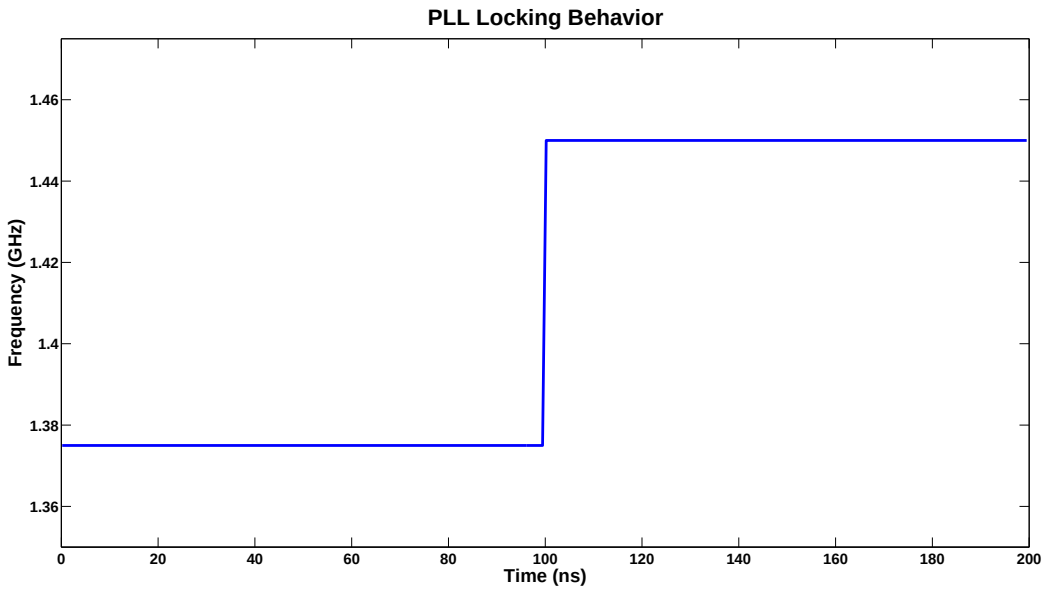


FIGURE 3.14. Input from ideal Frequency source.

offset. The phase noise of the created PLL system looks very good. The slight variation in phase noise can be ignored. Therefore, the phase noise will not considered as a metric in this design.

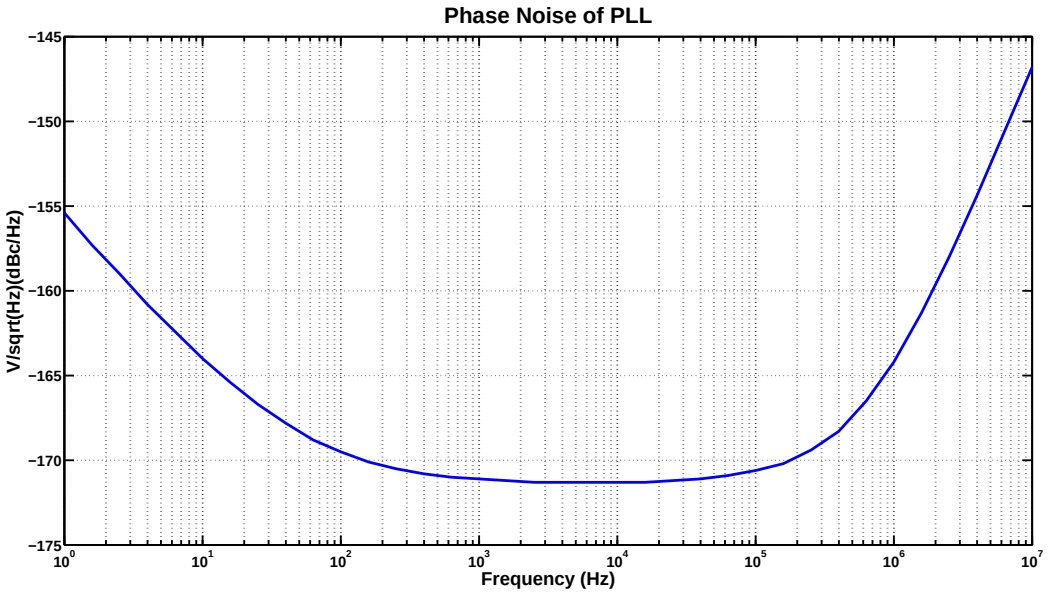


FIGURE 3.15. Phase noise of phase locked loop.

CHAPTER 4

PROPOSED METAMODEL-ASSISTED DESIGN FLOW

In order to speed up the design and optimization process of mixed-signal systems, new design flows are investigated in this dissertation. In this chapter, the proposed design flow is discussed from a high-level perspective. In order to get a comparative perspective, the traditional design flow that is used in practice is discussed as well.

4.1. Traditional Design Flow

AMS-SoC design is very complex because of the presence of heterogenous components in it. It involves multiple stages to reach the final product which is the physical design. Usually, a top-down approach is used for SoC design. The first step is to identify the AMS-SoC specifications. The specifications of the circuit and its operation environment play a major role in the design constraints and difficulty. The main stages of SoC design are shown in Figure 4.1.

After the system specifications are identified, the high level system design using algorithm based representation tools, such as MATLAB-Simulink is performed. This is accompanied by behavioral simulation using a high level modeling language such as, SystemVerilog, SystemC, VHDL, or Verilog. Mixed-signal systems as a whole are simulated using Verilog-AMS, VHDL-AMS, or SystemC-AMS. These tools enable designers to use multiple blocks of intellectual property cores (IP cores of digital and analog components) to simulate the system in fast and efficient simulation environment. These tools along with the IP cores can be used not only for design but also for verification purposes. These top-down approaches can also be used for software development well before the actual product is manufactured, reducing the design time of the product by enabling the design of hardware to be conducted concurrently with custom software.

As an example, SystemC, which is based on C++ with specific libraries, can simulate real-time circuit behavior in transaction level modeling. SystemC modeling includes digital logic, structural hierarchy and connectivity with clock cycle accuracy. The simulations from these platforms are not as accurate as register transfer level models. However, digital SystemC simulation is significantly faster and can provide speed up to simulation of roughly between 100-10,000$\times$.

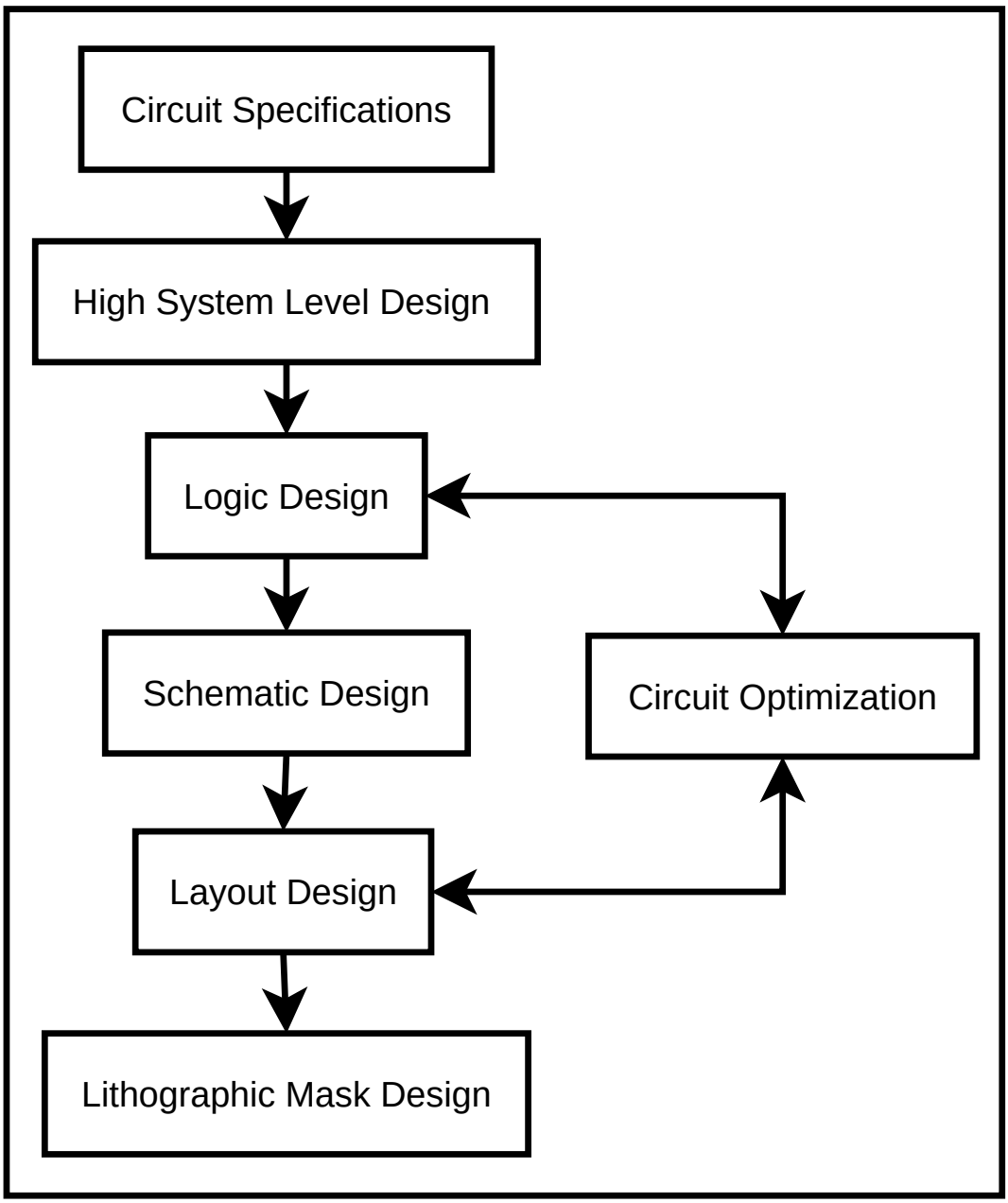


FIGURE 4.1. The traditional design flow for optimal physical design of mixed-signal system components.

This is due to the pin communication abstraction which is implemented as system calls within a language instead of using separate event triggers for each simulated block.

The next step for hardware design abstraction occurs as the register transfer level (RTL) languages. This is known as architectural level which is also part of high-level or behavioral level. Languages such as VHDL, Verilog, or Verilog-AMS, and VHDL-AMS, are used depending of

whether the SoC is digital or AMS-SoC. A mix of these languages can be used when performing component wise high-level or behavioral simulations.

Routing and placement of hardware components is one of the major issues for SoC design. Due to the large number of components on chip, logical design needs to correlate with area and time delay constraints, minimizing pin count and shortening wire length. The schematic design has to account for sizing components, especially in the analog domain, due to the large sizes of components like inductors and capacitors in comparison to transistor sizes for digital applications. Analog design usually is performed manually, while digital design can be conducted using standard cell libraries where most of the logic gates are standardized, which reduces design time. Manual labor is required for reducing critical path for timing consideration and optimization. Schematic design tools and simulators are used to help designers with simulations to verify that the calculations are correct. These tools are able to perform transient, spectrum, dc, analog and other types of analysis. The results are closer to silicon than other tools described earlier but they do not account for the parasitic components that are present in the system after the physical layout stage.

Once the logical design passes verification, the physical design stage is conducted. Automatic place and route algorithms have been developed for digital circuits. RF circuits are usually placed separately from digital since RF components can be very sensitive to the interference from the digital side of the SoC. Physical design has to be conducted according to the design rules described in the design manual for a targeted technology. Layout parasitic extraction extracts parasitic capacity and resistance from the layout database and outputs circuit diagrams to be used for simulation. Due to the increased number of components in the generated netlist, the simulation run time is very long but calculations on BSIM4/5 models are very close to actual silicon design [35]. Automatic layout design of the analog circuits or RF circuits is still an active research topic. Physical design of the analog or RF components of the AMS-SoC still involves tremendous manual labor. The time involved depends on the skill of layout engineers. In a typical design practice, multiple manual layout iterations are needed to obtain design closure [21]. A large number of layout iterations can lead to higher nonrecurring cost and overall chip cost. In addition, large number of layout iterations may also introduce design errors.

There are many metrics that need to be optimized during each phase of the design process. The optimization can also be performed at different levels of abstraction. The optimization objective depends on the target application. For most accurate results, the optimization needs to be conducted on the whole SoC. Furthermore, process variation and temperature deviation create more problems for the designers. Process variation is a statistical value that predicts deviation of die-to-die, wafer-to-wafer and lot-to-lot variations, while temperature affects transistor switching speeds and capacitance temperature instability. Usually Monte Carlo analysis or corner analysis are conducted to predict process and temperature variation. Various optimization algorithms are usually applied to find near optimal values for reduced process variation while performance of the circuit is within specifications. Each iteration of the optimization analysis is conducted by adjusting the parameters in the physical layout and running simulations, which is a very time consuming process. Therefore, the optimization stage usually does not reach optimal performance due to narrow time constraints of the design process. Optimization yields the parameter values for circuit devices, that in turn are used as the final design.

The optimized physical design is complete once all the above steps are followed. The design engineers then follow the steps for taping out the layout of the design. The lithography mask is generated for fabrication purposes. The design then is sent to the manufacturer for production. More testing is conducted at the plant and the design company to verify the device suitability for release to the customer.

4.2. Proposed Design Flow

This dissertation mostly targets the optimization stage of the design process. The proposed design flow depends on the following 3 aspects to achieve significantly fast design flow:

i. Performing optimization over the metamodels, not on the actual circuits.
ii. Using accurate and yet fast algorithms for optimization purposes.
iii. Using a maximum of two manual layout steps.
iv. Using automatic iterations over the metamodels.

As a result, the optimization time of the design will be reduced, which is one of the major time sinks in the traditional design flow.

The design flow that includes metamodels is proposed and is shown in Figure 4.2 [20]. The physical layout of the baseline design is parameterized and is sampled twice, once for training samples and another for verification. The metamodels are then created for each output set of the design. The metamodels are either polynomial or non-polynomial in nature. The type of metamodel depends on the target figure of merit and mixed-signal system components. More expensive optimization algorithms can be applied using metamodels due to their computational cheapness. The selection of appropriate algorithms for specific optimization is a matter of experience. At the end of the automatic iteration during the optimization process, the optimized values are obtained. The optimal parameters are then used to adjust the initial physical layout to create the near optimal design. This design flow only uses 2 iterations for the physical design, at the beginning and the end, minimizing the amount of time the designer need to spend on the circuit.

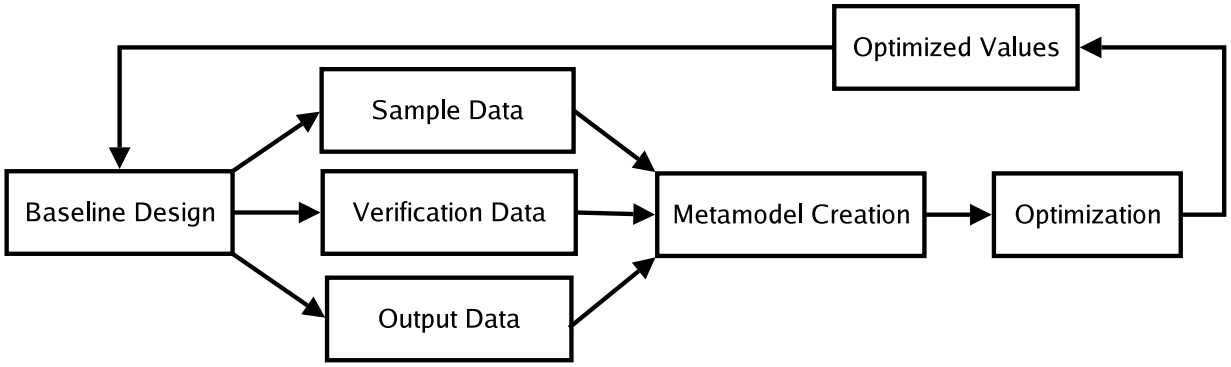


FIGURE 4.2. High-level representation of the metamodel-assisted ultra-fast design flow of AMS-SoC components.

Metamodels are proposed to assist and reduce the complexity of the simulations that are conducted on the physical layout. A metamodel can be described as a mathematical equation that simplifies the process and predicts the needed output. Metamodels are commonly used in various industries to predict the process behavior in computationally expensive or costly processes [19, 17]. There are numerous metamodels that can be generated from the same data set. Linear and nonlinear functions are used, but knowing which one to use requires the designer to pick the best one from the set of known equation datasets, such as, polynomial, Bayesian, spline, logarithmic,

sinusoidal, or any other known or custom functions. It is hard to find the proper function without knowing the behavior of the process itself. The right selection of the metamodel depends on the designer experience.

After the creation of the physical design, the design characteristic output values of the circuit change from the schematic due to additional parasitics. Usually comprehensive manual design iterations follow to adjust the physical design to bring the circuit back to the desired specifications. Then the statistical analysis (including Monte Carlo analysis or corner analysis) is followed to conduct process variation analysis. If the circuit does not pass the process variation specifications then more manual labor is needed to adjust the design and more simulations are required for this iterative flow.

The proposed design flow uses metamodels that can predict the characteristics of the circuit. The physical design parametric netlist is parameterized and sampling is used to create two data sets for training and verification of the metamodel. The metamodel is created based on the training data set and then verified for each characteristic of the circuit. More complex optimization algorithms can then be applied to the circuit using the metamodel. The simulations are not conducted at this stage and optimization therefore can be used for more iterations while still requiring significantly less time than actual simulations.

The proposed design flow with more detailed steps is shown in Figure 4.3. First he logical design is performed to meet the required specifications. The initial physical design is then implemented. The physical design is then subjected to Design Rule Check (DRC), Layout versus Schematic (LVS) verification, and parasitic (RCLK) extraction. If the specifications are not met, the parasitic netlist is then parameterized with design variables that can be considered in the circuit. The parameterized netlist is now the medium for automatic design iteration of the physical-design of the mixed-signal components. This netlist is then used by the automated process to create a metamodel by applying the best sampling technique described in this dissertation. Once the metamodel is created it can be optimized to find the parameters for the variables that were chosen before. The parameters that were generated in the optimization stage are then used to adjust the initial physical layout resulting in the final product.

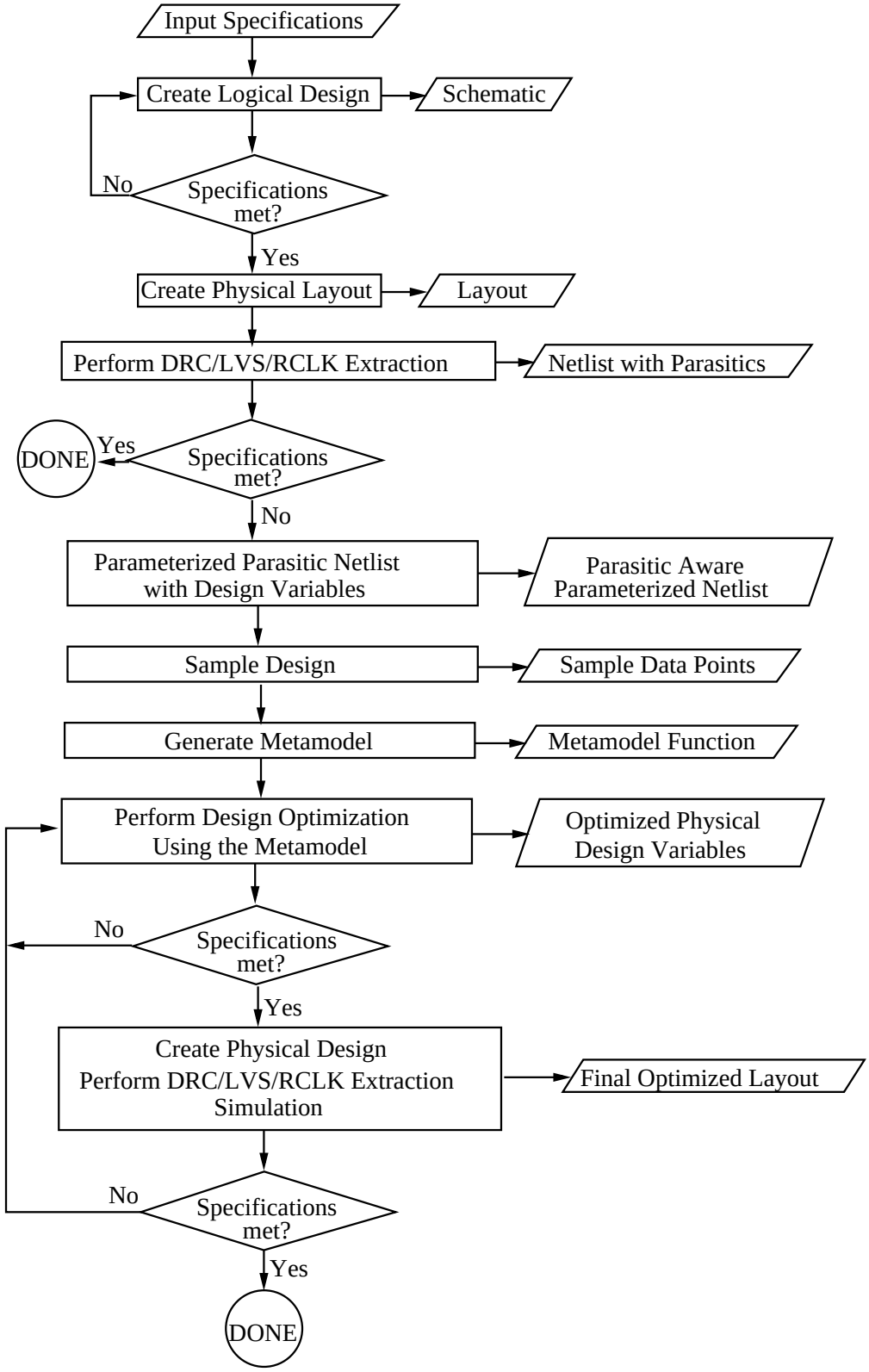


FIGURE 4.3. The proposed metamodel-based design flow. Speedup comes from the following: automatic iterations, use of the metamodels, and use of only two manual layout steps.

This design flow reduces the amount of manual labor down to two physical design iterations, considerably reducing the design process time, the initial design to sample the data and final design after the optimization of the metamodel. In this approach for this algorithm to work properly the key element is to create an accurate design in which the generation of a very accurate metamodel is essential. Hence this dissertation covers metamodel sampling techniques and their accuracy as shown in the outlined area in Figure 4.3.

CHAPTER 5

STATISTICAL TECHNIQUES FOR FAST MIXED-SIGNAL DESIGN SPACE SAMPLING FOR METAMODEL GENERATION

To create an accurate metamodel requires a sufficient number of samples. The larger the number of samples needed, the longer is the time, but the higher the accuracy. Thus, sampling time and sample points is a matter of tradeoff. To receive a single sample, one simulation run is required. In the proposed design flow the simulations are conducted on the parasitic-aware netlist of the physical design that take large amount of time. A good sampling technique can reduce the amount of samples, reducing the sampling time, while keeping the accuracy of the metamodel. This chapter discusses sampling techniques which are explored in the course of this research [17, 19]. Most common are the uniform, random and hybrid sampling techniques and they have been selected for comparison on the polynomial model that was created from the regression for power and frequency of two mixed signal circuits. Two case study circuits, a 45nm CMOS based ring oscillator and an 180nm LC-VCO are used for the purpose of experimentation.

5.1. Quality Metrics for Sampling and Metamodel Accuracy

Typically, the *actual* response of the circuit is unknown as there is a limited number of samples. However, since the picked test circuits are intentionally simple (to allow exhaustive sampling), an extremely accurate "golden" response surface with 10,000 sampling points can be used for validation and evaluation of the various metamodels. The "golden" response will be assumed as the true circuit response in the following discussion. The actual verification will probably use under 100 sample points in more complex circuits and substantially less points for very large circuits. The square root of mean square error (RMSE), shown in equation 16, is used to compare the prediction values of the metamodel generated using sampling points to the "true" response. RMSE estimates the difference between the metamodel and the true model. A smaller RMSE value identifies a more accurate metamodel on average [14]. Since we are also interested in overall accuracy of the metamodel, we calculated the standard deviation (σ) for all 10,000 points of the "true" response which is calculated as shown in equation 17. For a given RMSE a small σ value

shows that the metamodel has fewer values deviating from the mean error which otherwise could result in a large error in the constraint area of the metamodel.

The generation of the sampling points, SPICE runs and post-processing calculations are done automatically using a combination of commercial and in-house tools by using the following expressions:

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

$$\sigma = \sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(|y(W_{n_i}, W_{p_j}) - \hat{y}(W_{n_i}, W_{p_j})| - RMSE)^2}, \tag{17}$$

where (MN)=10,000 are points for each variation of W_n and W_p parameters that are selected in the design domain T for metamodel evaluation. These points are checked to ensure that they are not the same points used in the generation of the golden model. This would generate artificially small values of the RMSE. y and $\hat{y}$ are the responses at point W_{n_i}, W_{p_j} of the golden model and the metamodel, respectively.

All future data will be fitted into polynomial linear regression models in powers of four for the purpose of comparison. However, the DOE expression is fitted in powers of two due to the lack of sampling points in this case. Thus, the metamodel for two parameters (W_n for NMOS and W_p for PMOS has the following format:

$$y = \sum_{i,j=0}^{k} \left(\alpha_{ij} \times W_n^i \times W_p^j\right), \tag{18}$$

where y is the response being modeled (frequency in our case), W_n and W_p are variables and α_{ij} are the coefficients determined by the polynomial regression. Specifically, k =4 except in the case of DOE where k =2.

Uniform sampling is used to strategically sample the design space. Two different uniform sampling techniques are used: exhaustive sampling and design of experiments (DOE). DOE is used for highly dimensional design space sampling with a small number of samples for each dimension (2-3) and is sparse. At the same time, exhaustive sampling generates massive amount of samples

per dimension making it the most accurate and expensive model. As random sampling technique, Monte Carlo sampling is investigated.

5.2. Exhaustive Sampling

If simulation time is not an issue, the sampling could be used. However, in this dissertation exhaustive sampling is performed to generate the golden surface for the purpose of quality comparison. Exhaustive sampling is pure uniform sampling. With m as the number of variables and n as the number of runs for each variable the amount of samples it will take to create a metamodel will be n^m. The RMSE for a large amount of m is very small. In our case, taking in consideration the width of PMOS and NMOS as variables and running the simulation for these two variables 100 different times each, 10,000 simulation results are obtained for the given ring oscillator circuit. The RMSE for a metamodel of that amount of number of runs is minimal.

Figures 5.1 and 5.2 show the response surfaces of the frequency output on the z-axis with W_n and W_p on the x- and y-axis, accordingly for both circuits. Since the calculated RMSE is very small for this metamodel, it can be concluded that the generated metamodel data is the golden model for a comparison with the actual results for future simulations. Running this many simulations to receive an almost perfect metamodel is usually not practical in the design process. Therefore, other sampling techniques are explored in order to minimize the sampling amount for these two variables to receive the best metamodel that fits the design for the given circuit.

5.3. Design of Experiments (DOE)

Design of Experiments (DOE) is a technique that is most commonly used with a large number of variables to profile and generate the predictive function of the model. A three level DOE metamodel is created from (min,mid,max) points of each parameter. The metamodel can only be fitted using a 2nd degree polynomial function, instead of the 4th as used in the other examples, due to the small number of samples. The sampling is effective for high dimensional design space, giving a rough idea of the design behavior. The sampling is not able to predict non-linear circuit behavior, therefore the RMSE that was calculated for the DOE metamodel is considerably higher in comparison to the other techniques. The RMSE for the ring oscillator that was calculated for

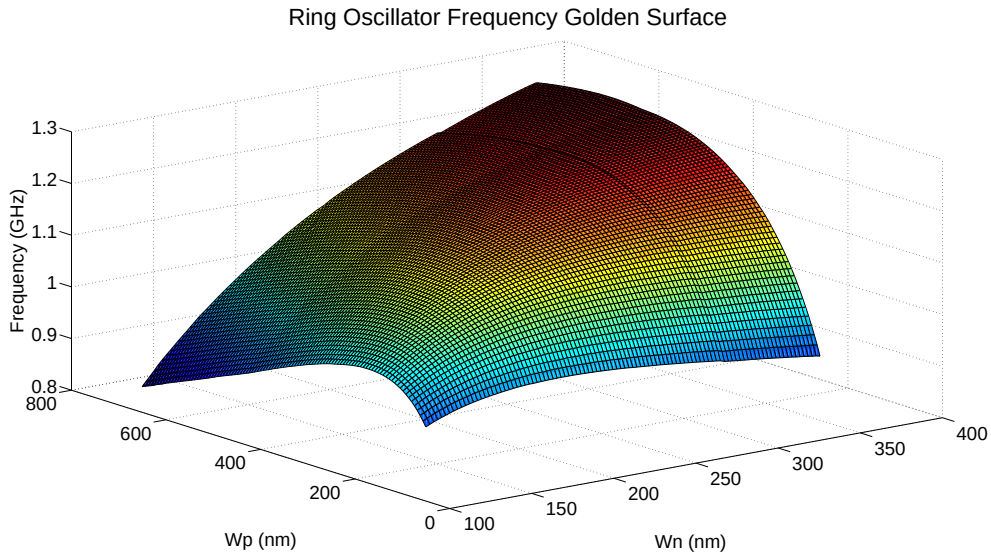


FIGURE 5.1. Exhaustive surface for the 45 nm ring oscillator circuit.

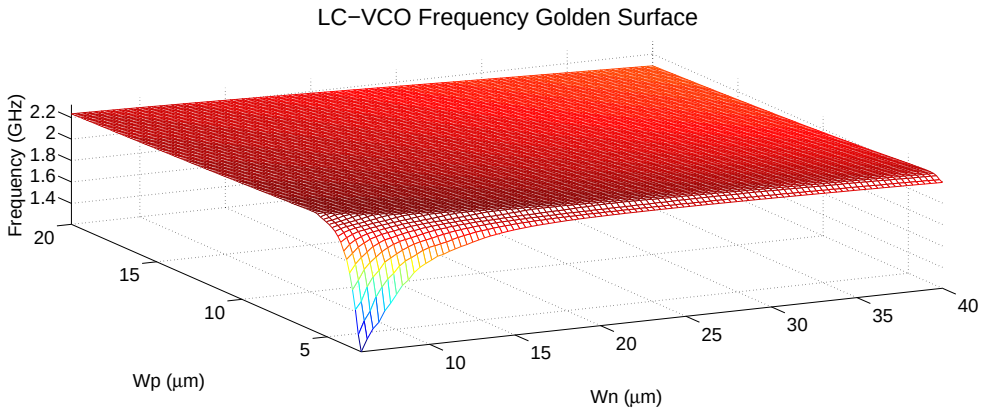


FIGURE 5.2. Exhaustive surface for the 180 nm LC-VCO circuit.

the DOE sample was 750 MHz with a standard deviation of 410 MHz. The highest variance for the error was 2.11 GHz. While the RMSE for the LC-VCO was 193.4 MHz and for the standard deviation was 219.4 Mhz. This data from both circuits indicate that DOE is not a competitive sampling technique for a small number of variables.

5.4. Latin Hypercube Sampling (LHS)

In 1975, W.J. Conover developed the stratified Monte Carlo sampling method which was called Latin hypercube sampling [29]. Latin hypercube sampling produces a random point within the generated n amount of Latin squares on the domain T. This technique provides more evenly

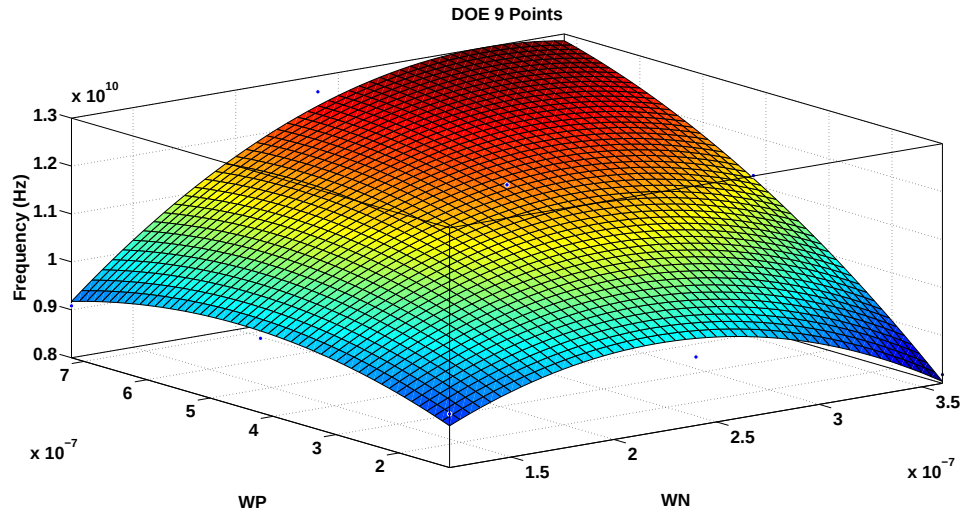


FIGURE 5.3. Surface for two dimensional sampling using DOE for the 45 nm ring oscillator circuit.

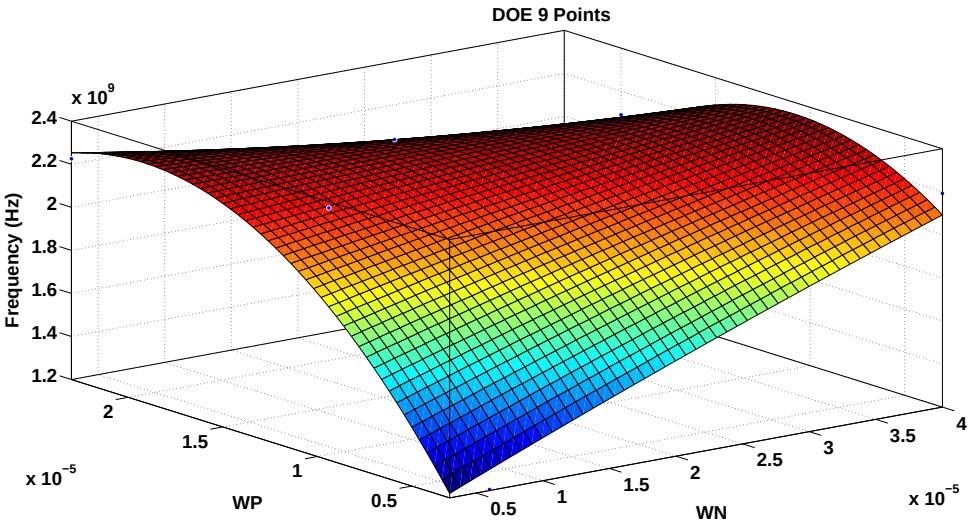


FIGURE 5.4. Surface for two dimensional sampling using DOE for the 180 nm LC-VCO circuit.

distributed sampling points than random sampling techniques, but the samples can still be clustered together as the samples are taken randomly from each Latin square as they can be adjacent to each other. Considering the same number of points as the Monte Carlo generated samples, Figure 5.5 shows the RMSE results for LHS metamodels for the ring oscillator circuit.

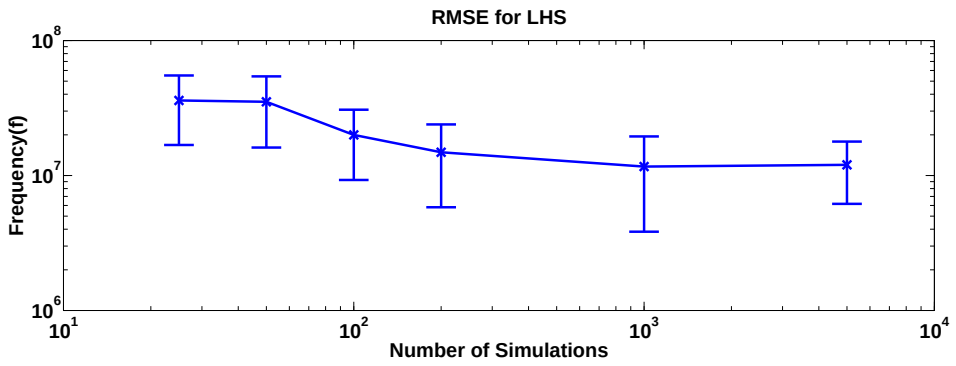


FIGURE 5.5. RMSE data for LHS of 45nm ring oscillator circuit.

LHS supports any number of planes and is proven to work better than Monte Carlo due to the more even distribution of points with still the random factor that helps to detect nonlinearity. LHS divides each plane (parameter) into Latin squares and using uniform distribution, randomly picks a point from each square. Output is generated for each run from SPICE simulation saving each needed value to its own data set. An example of LHS using 5000 samples on ring oscillator and 2000 samples on LC-VCO are shown in Figure 5.6 and Figure 5.7, respectively.

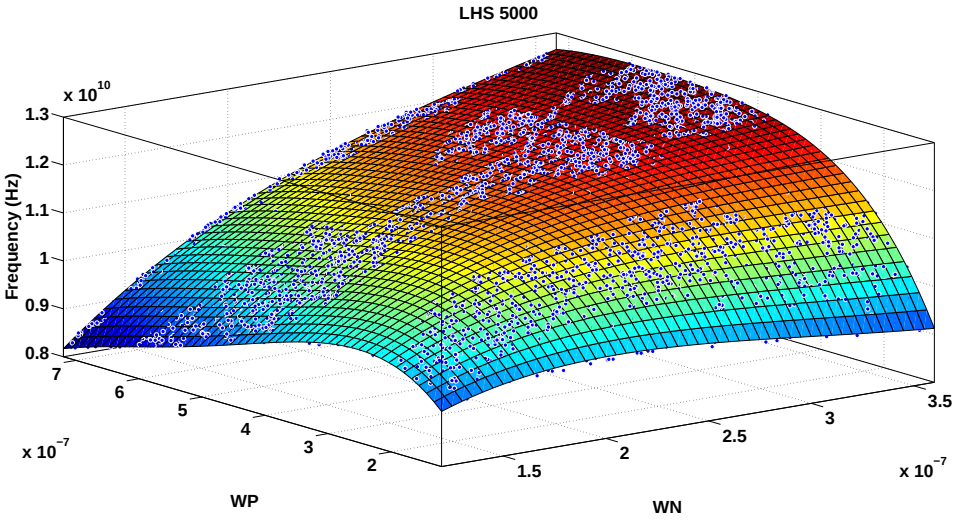


FIGURE 5.6. 5000 LHS sample points for the 45 nm ring oscillator circuit.

5.5. Middle Latin Hypercube Sampling (MLHS)

The Middle Latin Hypercube Sampling (MLHS) technique similar to regular LHS. It divides the domain T into n amount of Latin squares; however, instead of randomly sampling from

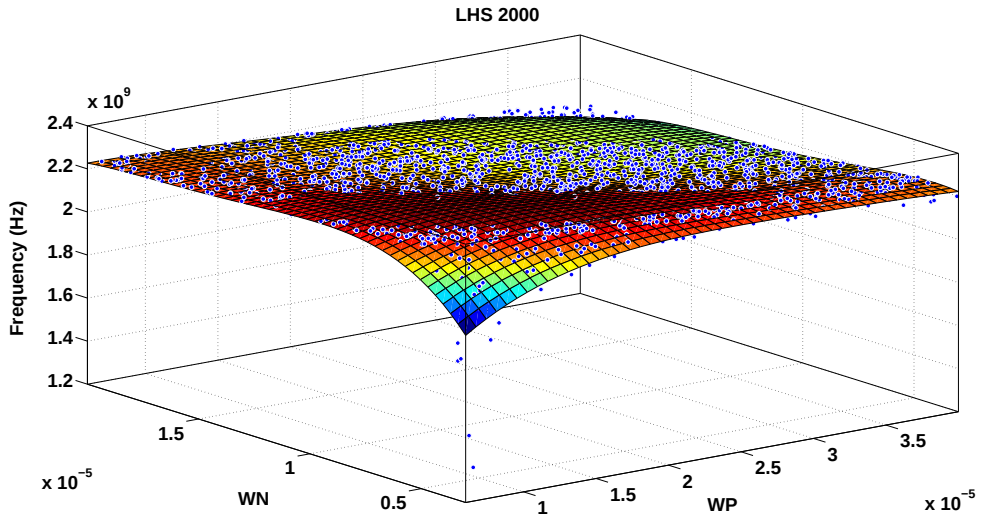


FIGURE 5.7. 2000 LHS sample points for the 180 nm LC-VCO circuit.

each of those squares, it picks the middle value from each one. This technique is more uniform than LHS. The main drawback is that it is not able to sample the regions close to the edge of the design space. Considering the same number of points as the Monte Carlo generated samples, Figure 5.8 shows the RMSE results for LHS metamodels for the 45 nm ring oscillator circuit.

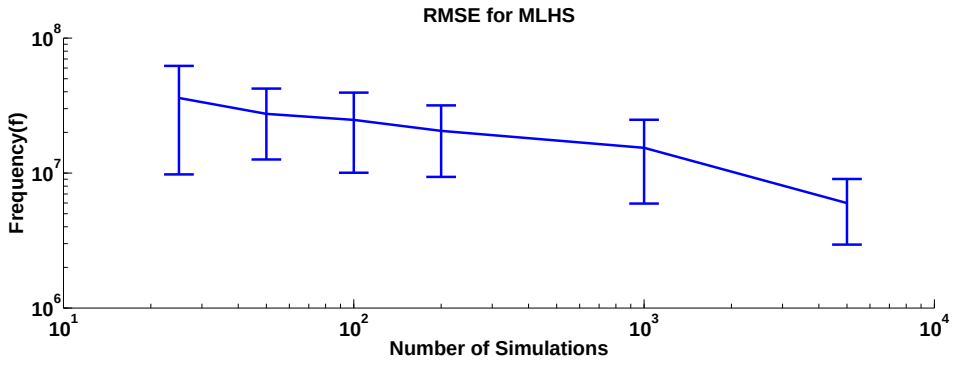


FIGURE 5.8. RMSE data for MLHS sampling of the 45 nm ring oscillator circuit.

5.6. The Monte Carlo Sampling

Monte Carlo or random sampling is a technique which samples the data for each variable, by picking n random data points with a given probability distribution with mean at the center of each variable in the design constraint domain T. Figure 5.13 shows the results for creation of

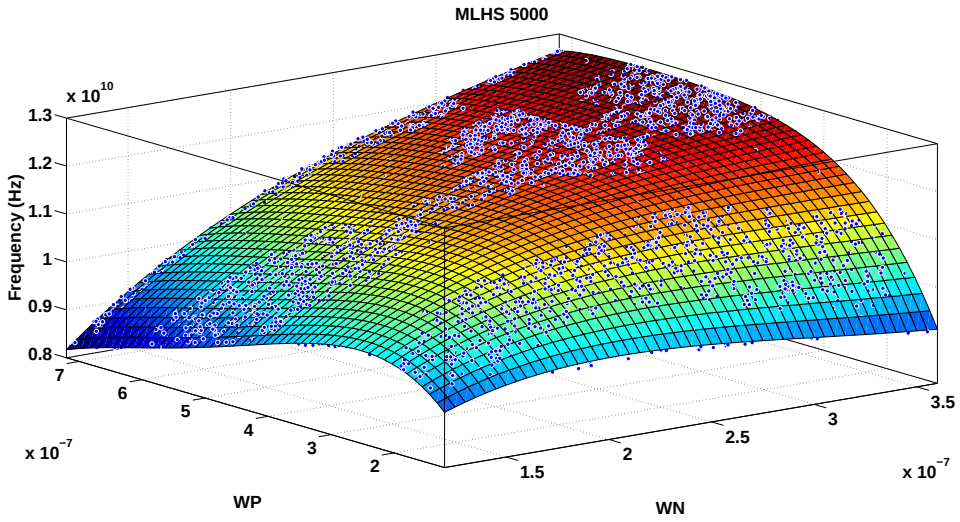


FIGURE 5.9. 5000 MLHS samples of the 45 nm ring oscillator circuit.

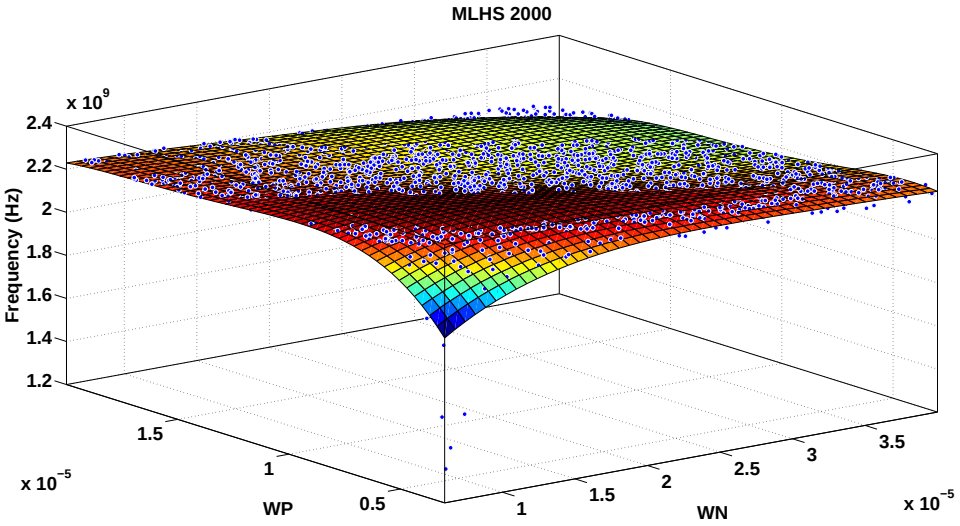


FIGURE 5.10. 2000 MLHS samples of the 180 nm LC-VCO circuit.

multiple metamodels with different number of samples and their RMSE results. Note that the RMSE and its standard deviation will both change each time the simulation is performed with the same amount of data points, since the data could have some areas of unsampled points or an over-abundant number of points in one area which is caused by the uneven distribution of sampling points in the domain T.

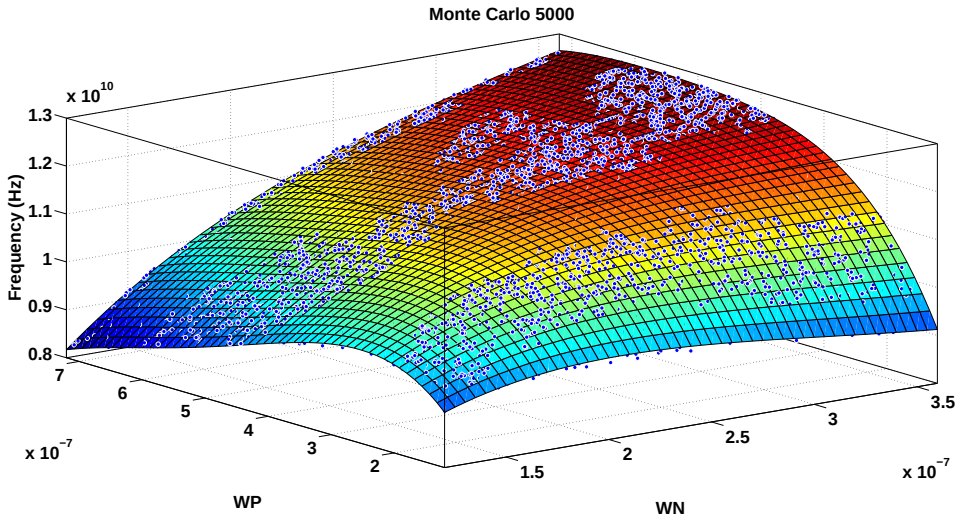


FIGURE 5.11. 5000 Monte Carlo simulation points for the 45 nm ring oscillator circuit.

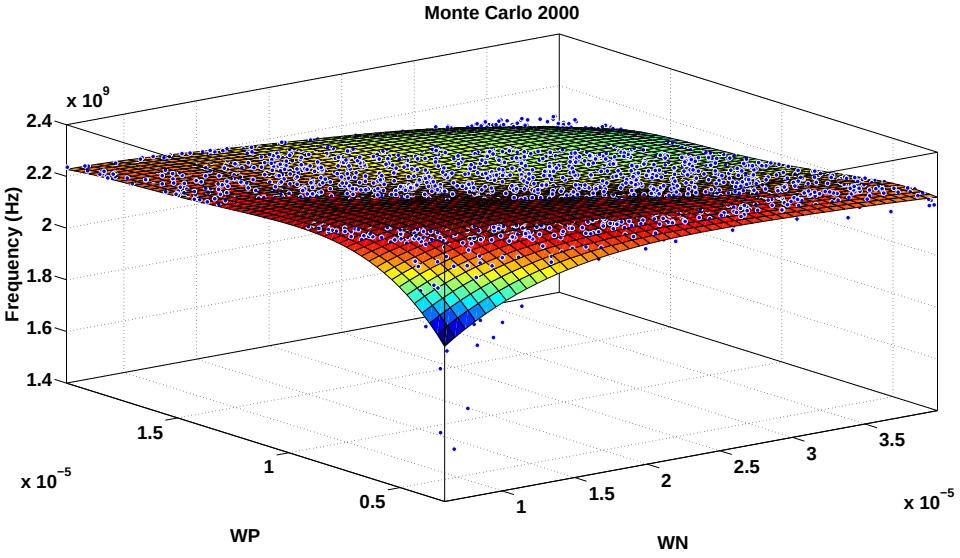


FIGURE 5.12. 2000 Monte Carlo simulation points for the 180 nm LC-VCO circuit.

5.7. Comparative Analysis of the Sampling Techniques

Monte Carlo sampling produces higher RMSE than uniform sampling because it is random and might not cover the full spectrum of its design variable. It is clear from the data provided in Tables 5.1 and Table 5.2 that uniform sampling provides superior accuracy to random sampling. Designers should choose LHS or MLHS over Monte Carlo. However, the trend in typical design environments is the opposite. This is probably due to the simplicity of running Monte Carlo versus

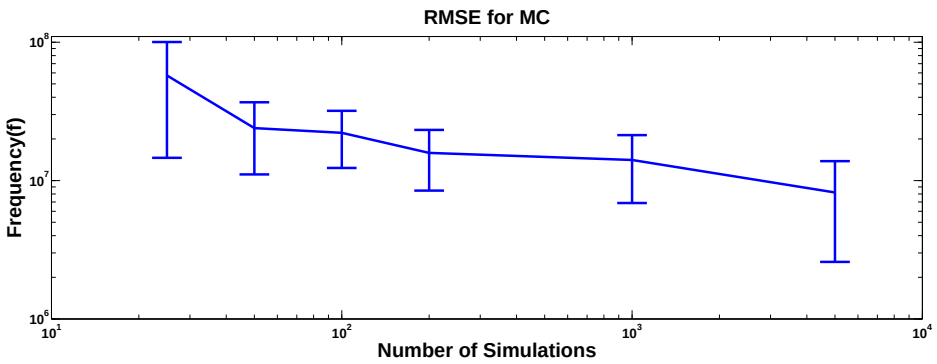


FIGURE 5.13. RMSE data for the 45 nm ring oscillator circuit Monte Carlo sampling. Note: the error bars have unequal lengths due to the logarithmic scale.

LHS or MLHS. Typically, most commercial simulators can perform Monte Carlo with a simple directive. On the other hand, uniform sampling requires extensive setup and if halted it takes extra effort to recover the design. However, the improved accuracy is well worth the extra effort.

The designer takes risks in picking Monte Carlo and LHS as MC and LHS property has random effect of sampled data. It can be seen in the LC-VCO data in Table 5.1 and Table 5.2 for Monte Carlo of 50 samples and LHS of 50 samples. The RMSE (μ) values are slightly higher for smaller number of samples due to the uneven distribution of points, but as the number of samples increases the distribution of samples does not have such a big effect on data.

Figure 5.14 and Figure 5.15 show the best results from multiple simulation runs for MC and LHS techniques in comparison to MLHS. It can be seen that the results are very close to each other, within 1% of the output frequency for the RO. Based on the average RMSE and standard deviation of errors (σ) data from Table 5.1 and Table 5.2, on average LHS performed slightly better than MC and MLHS for both circuits especially for low amount of samples. As the number of samples approach exhaustive sampling, the choice of sampling technique become irrelevant. The random effect for LHS and MC techniques provide a slight advantage to have a better fitted metamodel than MLHS. Based on these observations, the LHS is used for all further data sampling in this dissertation.

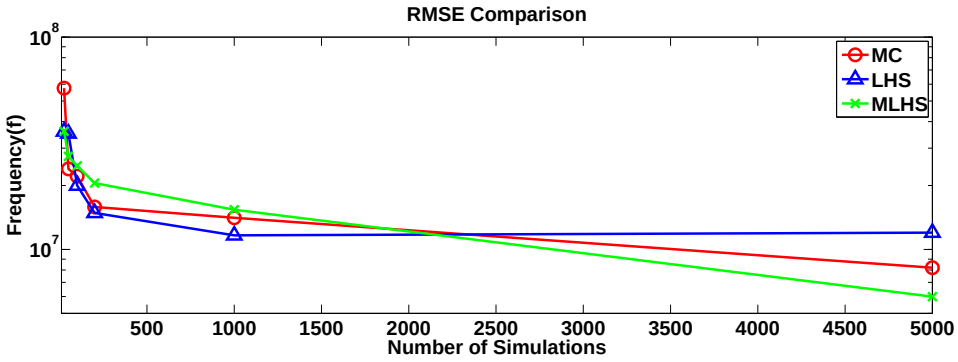
All LHS data is generated for further research in the following way. Matrix L is created to code each row as a term in the polynomial power for each parameter except that when the value is

TABLE 5.1. RMSE comparison for different sampling techniques for 45nm ring oscillator circuit (in MHz)

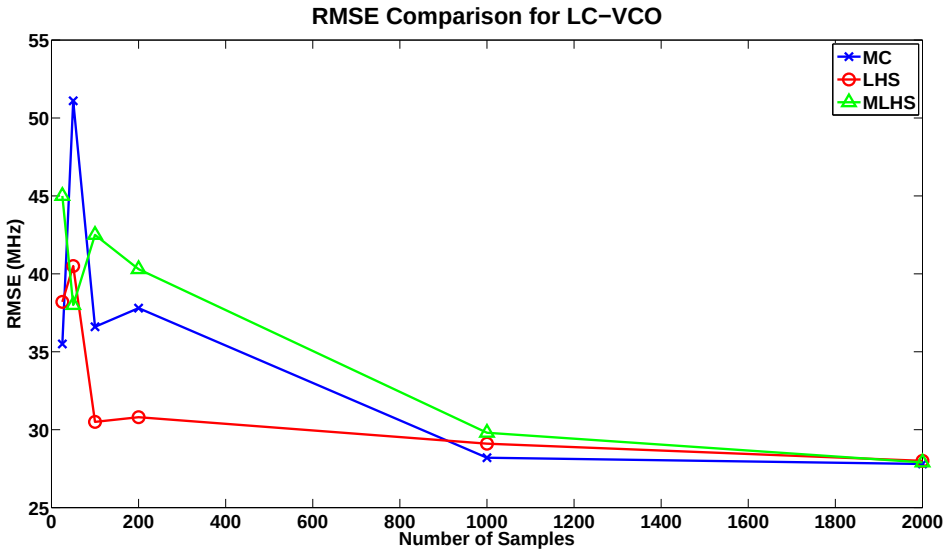
Samples	MC		LHS		MLHS	
Ring Oscillator						
N	μ	σ	μ	σ	μ	σ
25	57.5	42.9	35.6	19.1	36.0	26.2
50	24.0	12.9	35.2	19.1	27.4	14.8
100	22.1	9.7	20.0	10.7	24.8	14.7
200	15.9	7.3	14.9	9.0	20.5	11.2
1000	14.1	7.2	11.7	7.8	15.4	9.4
5000	8.2	5.6	12.0	5.8	5.9	3.0
Average	23.6	14.3	21.6	11.9	21.7	13.2

TABLE 5.2. RMSE comparison for different sampling techniques for 180nm LC-VCO circuit (in MHz)

Samples	MC		LHS		MLHS	
LC-VCO						
N	μ	σ	μ	σ	μ	σ
25	35.5	33.4	38.2	36.7	45.0	39.7
50	51.1	41.9	40.5	39.9	38.0	31.9
100	36.6	28.9	30.5	29.3	42.5	38.8
200	37.8	32.7	30.8	36.8	40.3	36.2
1000	28.2	25.9	29.1	26.3	29.8	24.8
2000	27.8	25.3	28.0	24.9	27.9	24.6
Average	36.2	31.4	32.9	32.3	37.3	32.7



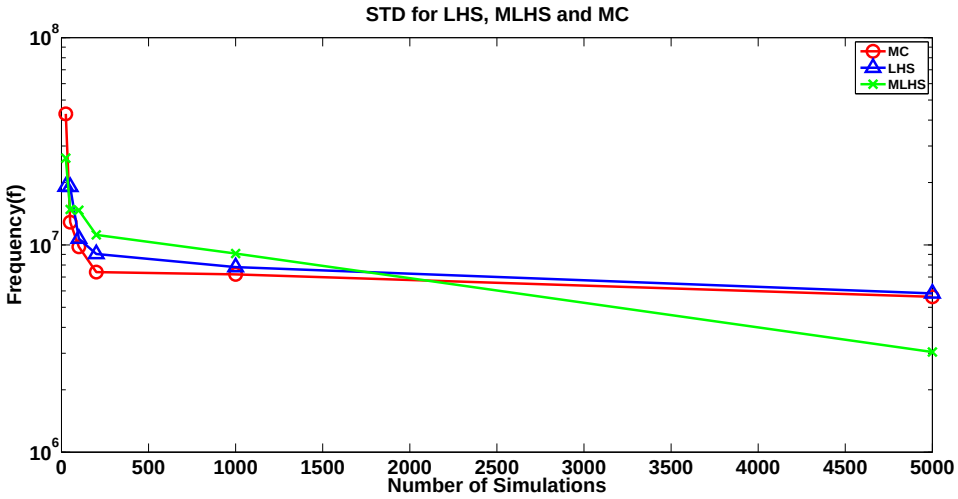
(a) RMSE comparison for 45 nm ring oscillator circuit.



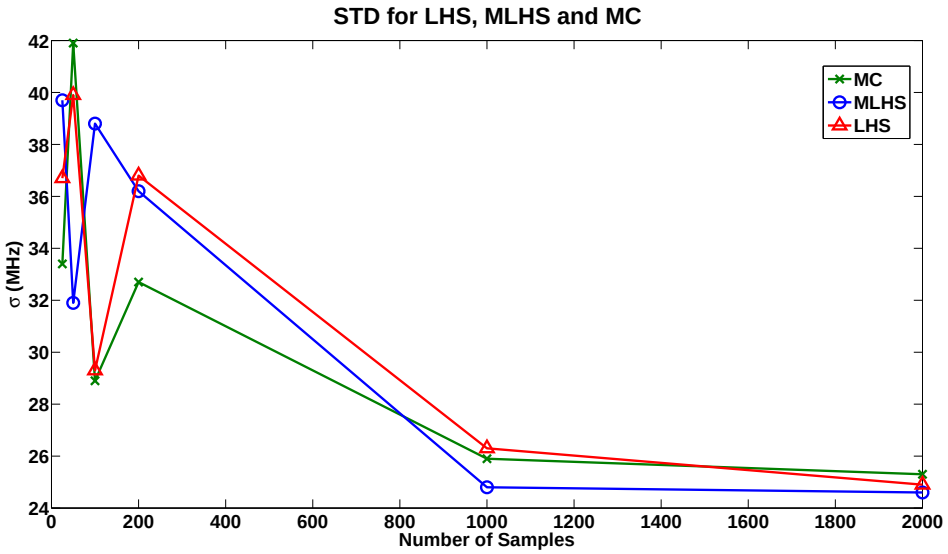
(b) RMSE comparison for the LC-VCO.

FIGURE 5.14. RMSE comparison for both ring oscillator and LC-VCO circuits.

0, it is implied that the term is not present. As an example L is the matrix for second order code for 2 parameters.



(a) STD comparison for the 45 nm ring oscillator circuit.



(b) STD comparison for the 180 nm LC-VCO circuit.

FIGURE 5.15. STD comparison for both ring oscillator and LC-VCO circuits

(19)
$$L = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 2 & 0 \\ 0 & 1 \\ 0 & 2 \\ 1 & 1 \end{bmatrix}$$

Lets assume that matrix $X = [p_1, p_2]$ where p_1, p_2 are parameters. Then the design matrix template (DMT) for X using L code becomes $DMT(X, L) = [0, p_1, {p_1}^2, p_2, {p_2}^2, p_1 p_2]$. The Design Matrix (DM) is created from using LHS of p_1 and p_2 parameters, resulting in the matrix $DM^T(X, L) =$

$$\begin{bmatrix} 0 & \dots & 0 \\ lhs(p_1)_1 & \dots & lhs(p_1)_n \\ lhs(p_1)_1^2 & \dots & lhs(p_1)_n^2 \\ lhs(p_2)_1 & \dots & lhs(p_2)_n \\ lhs(p_2)_1^2 & \dots & lhs(p_2)_n^2 \\ lhs(p_1)_1 lhs(p_2)_1 & \dots & lhs(p_1)_n lhs(p_2)_n \end{bmatrix} \tag{20}$$

The corresponding Y matrix is then sampled from the original design for target figure of merit (FoM) and is based on the parameters p_1 and p_2 for each row in the DM matrix.

CHAPTER 6

POLYNOMIAL AND NEURAL NETWORK BASED METAMODELING

The core contribution of this dissertation is the metamodeling of mixed-signal system components. In this chapter the metamodels are discussed [19, 18, 20, 17]. The metamodels are either polynomial or non-polynomial. This chapter discusses neural network based non-polynomial metamodels. Relevant metrics are presented to quantify the quality of the metamodels. As the metamodels are generated using full-blown parasitic-aware netlists, they are silicon accurate.

6.1. General Perspective of Metamodeling

To create a metamodel the designer needs to answer the following questions first. How many parameters are in the circuit, and how many samples need to be performed to create the metamodel. The sampling stage is the slowest part of the design process. The accuracy of the metamodel is dependent on the amount of simulations as it also corresponds to the maximum number of coefficients that can be fit in the model. To maximize the accuracy of the model, the right sampling technique needs to be used. In Chapter 5 we have shown that Latin hypercube sampling (LHS) is the most accurate sampling technique. The right fitting function can also affect the accuracy. This chapter investigates different fitting functions and models.

There are many options to fit the sampled data using many different functions. Typically, it is not a very time consuming process. For large circuits like the PLL, the data can be fit into a partial polynomial equation. Since the full polynomial function would result in a very large number of coefficients for 21 selected variables, partial polynomial functions using stepwise fitting in order of 1 through 6 are considered. The final outcome is a multivariate polynomial function in the form of the following expression:

$$\hat{y} = \sum_{i=0}^{d} a_{i_1 i_2 \ldots i_d} \prod_{j=1}^{d} x_j^{i_j}, \tag{21}$$

where the response $\hat{y}$ is the FoM output for the given sampling points.

There may be numerous metamodels created from the same sampled data points. After the metamodels are created, the right metamodels need to be selected for the use in the context of

the design flow. Root mean square error (RMSE) and R^2 are the metrics used for measuring the goodness of fit. Root mean square error (RMSE) is derived from the sum of square errors (SSE) using the following expressions:

$$RMSE = \sqrt{\frac{1}{N}SSE}, \tag{22}$$

$$= \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(y(x_k)-\hat{y}(x_k)\right)^2}, \tag{23}$$

where y are the actual simulation result values and $\hat{y}$ are the results of the metamodel at the same location as the simulation point.

The coefficient of determination R^2 predicts the probability of the future result to be predicted by the model. It is also used to verify the accuracy of the metamodels. R^2 ranges from 0-1 where 1 is the best value. Essentially R^2 is calculated using the following formula:

$$R^2 = \frac{SS_{err}}{SS_{tot}}, \tag{24}$$

$$= \frac{\sum_i \left(y_i - f_i\right)^2}{\sum_i \left(f_i - \bar{y}\right)^2}. \tag{25}$$

However, R^2 cannot account for over-fitting of the model. Therefore, the R^2_{adj} accounts for the number of explanatory terms in a models which is presented using the following expression:

$$R^2_{adj} = \left(1 - \frac{SS_{err}}{SS_{tot}}\frac{df_t}{df_e}\right). \tag{26}$$

6.2. Polynomial Metamodeling

The creation of the metamodel is the crucial step in this design flow whose quality is affected by design space sampling and the nature of mathematical functions in the metamodel. As it is not possible to exhaustively sample all the points with large amount of parameters, the accuracy of the model varies for different forms of the metamodel. In the polynomial metamodels, polynomial functions in the power of one through six are used. However, in general, this process can be extended to other functions.

Partial polynomial functions are generated using the stepwise regression method which filters out the coefficients from a full polynomial equation. The filtering essentially drops the

coefficients that do not contribute to the outcome of the function. This method results in shorter and sufficiently accurate functions with faster computation time. Stepwise regression is a systematic method for adding and removing terms from a model based on their statistical significance in a regression.

Stepwise regression starts with an initial model and then compares the explanatory power of incrementally larger and smaller models. At each step, the p-value of an F-statistic is computed to test the model with and without a potential term. If a term is not currently in the model, the null hypothesis is that the term would have a zero coefficient if added to the model. If there is sufficient evidence to reject the null hypothesis, the term is added to the model otherwise. If a term is currently in the model, the null hypothesis is that the term can be ignored. If there is insufficient evidence to reject the null hypothesis, the term is removed from the model. The method finalizes when there no more improvements can be done to the model. Stepwise regression may build different models from the same set of potential terms depending on the terms that were initially included in the model which changes the order in which terms are moved in and out. To make this process robust the following flow for metamodel generation has been described in algorithm 1.

Algorithm 1 The Metamodel Generation Algorithm.

```
sample = generate_sample_data().
verify = generate_verify_data().
centered = center(sample).
for degree = 1 to 6 do
    sample_mm(degree)=stepwise(sample, degree).
    centered_mm(degree)=stepwise(centered, degree).
    sample_RMSE(degree)=verify(sample_mm, verify).
    centered_RMSE(degree)=verify(centered_mm, verify).
end for
pick_lowerst(sample_RMSE, centered_RMSE).
```

R^2 and R^2_{adj} for different orders of the polynomial metamodel are shown in Figure 6.1. Figure 6.2 shows the number of coefficients that are generated for each order of the polynomial

metamodel. In that figure the R^2 value and R^2_{adj} nearly equal to 1 when the order reaches 5. The number of coefficients that represent the function at those orders does not change. That means that the model is over fitted. Therefore, for the metamodel that represents settling time, a polynomial order of 4 will be used.

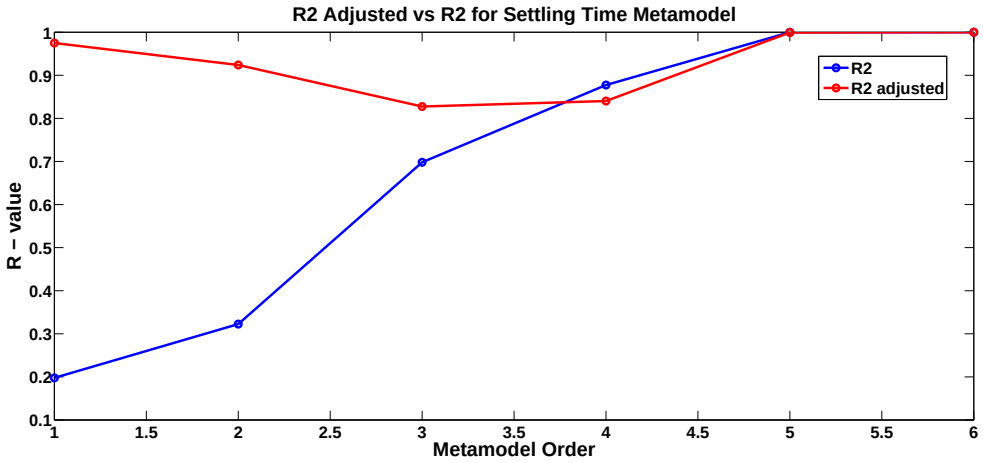


FIGURE 6.1. Generated R^2 and R^2_{adj} for every order of the polynomial metamodel for settling time.

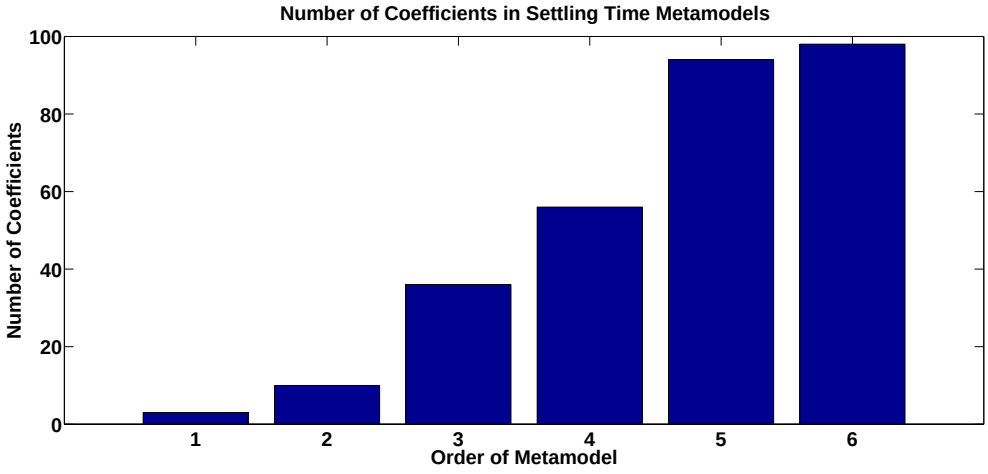


FIGURE 6.2. Number of coefficients corresponding to the order of the generated metamodel for settling time.

Next, the metamodels in first to fifth orders of polynomial are compared. The sampling technique that was used for metamodel creation was LHS, as it has performed the best, as discussed earlier. During the design process it is necessary to allocate time for sampling the data and

verification data to find out how accurate is the metamodel. Typically, only around 30% from sampled data needs to be allocated for verification process. The metamodeling generation results for the 45 nm ring oscillator circuit and the 180 nm LC-VCO circuit are shown in Table 6.1. For both cases, LHS with only 100 sample points are used. The error distribution is also shown to compare the metamodels.

TABLE 6.1. Metamodel polynomial order comparison for the 45 nm ring oscillator and the 180 nm LC-VCO circuits (in MHz) on 100 samples.

Circuit	Order	μ error	σ error
Ring Oscillator	1	571.0	286.7
	2	195.4	78.1
	3	37.2	18.0
	4	20.0	10.7
	5	17.1	9.6
LC-VCO	1	42.3	40.1
	2	39.4	37.8
	3	35.4	33.9
	4	30.5	29.3
	5	26.5	25.2

From the data that is presented in Table 6.1 it is observed that the higher polynomial functions show best results. This is happening for both of the circuits investigated. The generation of metamodels to perform linear regression only takes seconds so the generation times are not provided. The following polynomial functions have been generated for Table 6.1.

Ring oscillator Order 1:

$f(W_n, W_p) = 7.94 \times 10^9 + 1.1 \times 10^{16} W_n + 1.28 \times 10^{15} W_p$.

Ring oscillator Order 2:

$f(W_n, W_p) = 6.38 \times 10^9 + 2.2 \times 10^{16} W_n + 6.1 \times 10^{15} W_p - 5.03 \times 10^{22} W_n^2 + 3.28 \times 10^{22} W_n W_p - 1.52 \times 10^{22} W_p^2$.

Ring oscillator Order 3:

$f(W_n, W_p) = 4.71 \times 10^9 + 3.9 \times 10^{16} W_n + 1.26 \times 10^{16} W_p - 1.7 \times 10^{23} W_n^2 + 8.84 \times 10^{22} W_n W_p - 5.14 \times 10^{22} W_p^2 + 1.85 \times 10^{29} W_n^3 + -3.77 \times 10^{28} W_n^2 W_p - 4.77 \times 10^{28} W_n W_p^2 + 3.83 \times 10^{28} W_p^3.$

Ring oscillator Order 4:

$f(W_n, W_p) = 3.55 \times 10^9 + 4.47 \times 10^{16} W_n + 2.41 \times 10^{16} W_p - 2.52 \times 10^{23} W_n^2 + 1.45 \times 10^{23} W_n W_p + -1.19 \times 10^{23} W_p^2 + 4.69 \times 10^{29} W_n^3 - 1.15 \times 10^{29} W_n^2 W_p - 1.46 \times 10^{29} W_n W_p^2 + 1.75 \times 10^{29} W_p^3 - 2.9 \times 10^{35} W_n^4 - 5.63 \times 10^{34} W_n^3 W_p + 1.54 \times 10^{35} W_n^2 W_p^2 + 2.08 \times 10^{34} W_n W_p^3 - 8.54 \times 10^{34} W_p^4.$

Ring oscillator Order 5:

$f(W_n, W_p) = 2.19 \times 10^9 + 8.04 \times 10^{16} W_n + 2.24 \times 10^{16} W_p - 6.73 \times 10^{23} W_n^2 + 2.82 \times 10^{23} W_n W_p - 1.62 \times 10^{23} W_p^2 + 2.44 \times 10^{30} W_n^3 - 4.06 \times 10^{29} W_n^2 W_p - 4.87 \times 10^{29} W_n W_p^2 + 3.66 \times 10^{29} W_p^3 - 4.42 \times 10^{36} W_n^4 - 1.49 \times 10^{35} W_n^3 W_p + 9.54 \times 10^{35} W_n^2 W_p^2 + 2.53 \times 10^{35} W_n W_p^3 - 3.62 \times 10^{35} W_p^4 + 3.3 \times 10^{42} W_n^5 + 3.94 \times 10^{41} W_n^4 W_p - 3.24 \times 10^{41} W_n^3 W_p^2 - 4.46 \times 10^{41} W_n^2 W_p^3 - 5.97 \times 10^{39} W_n W_p^4 + 1.34 \times 10^{41} W_p^5.$

LC-VCO Order 1:

$f(W_n, W_p) = 2.38 \times 10^9 - 3.49 \times 10^{12} W_n - 6.66 \times 10^{12} W_p.$

LC-VCO Order 2:

$f(W_n, W_p) = 2.3 \times 10^9 + 6.25 \times 10^{11} W_n + 1.45 \times 10^{11} W_p - 4.16 \times 10^{16} W_n^2 - 2.01 \times 10^{17} W_n W_p - 1.02 \times 10^{17} W_p^2.$

LC-VCO Order 3:

$f(W_n, W_p) = 2.06 \times 10^9 + 1.58 \times 10^{13} W_n + 4.72 \times 10^{13} W_p - 4.66 \times 10^{17} W_n^2 - 1.39 \times 10^{18} W_n W_p - 3.37 \times 10^{18} W_p^2 + 4.48 \times 10^{21} W_n^3 + 9.75 \times 10^{21} W_n^2 W_p + 3.33 \times 10^{22} W_n W_p^2 + 7.23 \times 10^{22} W_p^3.$

LC-VCO Order 4:

$f(W_n, W_p) = 1.49 \times 10^9 + 6.34 \times 10^{13} W_n + 2.05 \times 10^{14} W_p - 2.44 \times 10^{18} W_n^2 - 7.86 \times 10^{18} W_n W_p - 2.08 \times 10^{19} W_p^2 + 4.39 \times 10^{22} W_n^3 + 1.41 \times 10^{23} W_n^2 W_p + 3.74 \times 10^{23} W_n W_p^2 + 9.29 \times 10^{23} W_p^3 - 3.02 \times 10^{26} W_n^4 - 1.05 \times 10^{27} W_n^3 W_p - 2.59 \times 10^{27} W_n^2 W_p^2 - 6.52 \times 10^{27} W_n W_p^3 - 1.54 \times 10^{28} W_p^4.$

LC-VCO Order 5:

$f(W_n, W_p) = 4.06 \times 10^8 + 1.72 \times 10^{14} W_n + 6.13 \times 10^{14} W_p - 8.32 \times 10^{18} W_n^2 - 3.12 \times 10^{19} W_n W_p - 8.46 \times 10^{19} W_p^2 + 2.16 \times 10^{23} W_n^3 + 8.95 \times 10^{23} W_n^2 W_p + 2.34 \times 10^{24} W_n W_p^2 + 5.97 \times 10^{24} W_p^3 - 2.94 \times$

$10^{27}W_n^4 - 1.3 \times 10^{28}W_n^3W_p - 3.72 \times 10^{28}W_n^2W_p^2 - 8.3 \times 10^{28}W_nW_p^3 - 2.12 \times 10^{29}W_p^4 + 1.66 \times 10^{31}W_n^5 + 7.47 \times 10^{31}W_n^4W_p + 2.36 \times 10^{32}W_n^3W_p^2 + 5.41 \times 10^{32}W_n^2W_p^3 + 1.17 \times 10^{33}W_nW_p^4 + 2.98 \times 10^{33}W_p^5$.

The RMSE value that is calculated from metamodel fitting for sampling points is not a good metric for the metamodel fit. Thus, it is important to calculate RMSE of the metamodel versus the verification samples. Extra verification points are required to make sure that the fit is also good at other points than the ones being sampled. Verification samples are checked to make sure that the data points are not the same as the sampling points. The lowest RMSE value for different metamodel functions is then selected as the best function for use in design flow.

A total of 12 polynomial metamodels are generated. Out of the 12 polynomial metamodels of charge pump, 6 metamodels are using the sample points with data centering and 6 metamodels are without using the data centering. All the polynomial metamodels are then compared for accuracy. The lowest RMSE verification value is used to pick the most accurate metamodel. The generated polynomial metamodels will later be used in Chapter 7.

6.3. Neural Network Metamodeling

Neural network models are composed of a mass of fairly simple computational elements and rich interconnections between them. Neural networks operate in a parallel and distributed fashion which may resemble biological neural networks. Most of the neural networks have some sort of "training" rule in which the weights of connections are adjusted on the basis of presented patterns. Neural networks normally have great potential for parallelism, as the computations of the components are independent of each other. It has been proven in the universal approximation theorem that a neural network with one hidden layer can estimate any continuous function that maps to real numbers.

A multiple layer neural network consists of input, with nonlinear activation function in hidden layer, and linear activation function in the output layer. This makes multilayer networks very flexible and powerful due to their ability to represent nonlinear as well as linear functions. The multilayer network needs to have at least one non-linear function otherwise a composition of

TABLE 6.2. Comparison of different degree dolynomial detamodels for PLL component circuits.

Component	Degree	R^2	R^2 adjusted	Coefficients total	Coefficients in model	RMSE fit (W)	RMSE verification (W)
Divider	2	0.3503	0.9901	55	16	7.3×10^{-6}	5.9×10^{-6}
	3	0.4504	0.9618	220	66	6.9×10^{-6}	5.8×10^{-6}
	4	0.6787	0.8828	715	268	6.0×10^{-6}	9.1×10^{-6}
	5	1	1	2002	999	0	1.1×10^{-4}
LC-VCO	2	0.9769	0.9999	6	6	2.5×10^{-5}	2.1×10^{-5}
	3	0.9878	0.9999	11	11	1.8×10^{-5}	1.7×10^{-5}
	4	0.9927	0.9999	16	16	1.4×10^{-5}	1.4×10^{-5}
	5	0.9942	0.9999	21	20	1.2×10^{-5}	1.2×10^{-5}
	6	0.9946	0.9999	28	20	1.2×10^{-5}	1.1×10^{-5}
Charge Pump with data centering	2	0.9968	1	15	14	1.3×10^{-6}	8.4×10^{-4}
	3	0.9987	1	35	31	8.6×10^{-7}	8.5×10^{-4}
	4	0.9995	1	70	54	5.3×10^{-7}	8.4×10^{-4}
	5	0.9999	1	126	87	2.8×10^{-7}	8.2×10^{-4}
	6	1	1	210	151	1.4×10^{-7}	8.3×10^{-4}
Charge Pump without data centering	2	0.9968	1	15	14	1.3×10^{-6}	3.0×10^{-5}
	3	0.9987	1	35	30	8.6×10^{-7}	7.6×10^{-2}
	4	0.9995	1	70	41	5.6×10^{-7}	2.6
	5	0.9999	1	126	87	2.9×10^{-7}	1.2×10^{2}
	6	1	1	210	98	1.6×10^{-7}	4.4×10^{3}
Phase Detector	2	0.9603	0.9993	28	18	1.1×10^{-10}	7.1×10^{-10}
	3	0.9663	0.9989	84	34	1.0×10^{-10}	7.0×10^{-10}
	4	0.9748	0.9974	210	94	9.2×10^{-11}	7.1×10^{-10}
	5	0.9926	0.9955	462	206	8.2×10^{-11}	7.1×10^{-10}
	6	0.9968	0.9938	924	662	5.4×10^{-11}	9.7×10^{-10}

linear functions becomes just another linear function. The functions are of the following format:

$$\hat{y} = \sum_{j=1}^{d} \beta_j b_j(v_j) + \beta_0. \tag{27}$$

The linear layer function follows the format below:

$$v_i = \sum_{i=1}^{s} w_{ji} x_i + w_{j0}, \tag{28}$$

where w_{ji} is the weight connection between the jth component in the hidden layer and the ith component of the input. The nonlinear $\tanh$ activation function used for hidden layer was shown to work best for the PLL circuit:

$$b_j(v_j) = \tanh(\lambda v_j). \tag{29}$$

The network training is performed to minimize the least square criterion:

$$E = \sum_{k=1}^{n} (y_k - \hat{y_k})^2. \tag{30}$$

The weights between the input and hidden layers are updated as follows:

$$\Delta w_{ji} = \rho(y_k - \hat{y_k})\beta_j \left(\frac{\delta b_j}{\delta v_j}\right) x_i. \tag{31}$$

The input data set is generated from the SPICE simulations of the parasitic-aware netlist. It is the same for every metamodel and it is generated using Latin hypercube sampling (LHS). LHS supports any amount of planes and is proven to work better than Monte Carlo due to the more even distribution of points with still the random factor that helps to detect nonlinearity. LHS divides each plane (parameter) into Latin squares and randomly picks a point from each square. Output is generated for each run from SPICE simulation saving each needed value to its own data set. Hence, each metamodel will have its own target data set. This dissertation targets neural networks that have single output with multiple inputs. Therefore for future use of the neural network they can be used as functions, due to their format.

As the input data set has some values for parameters in different power of ten, it is important to either normalize or standardize the input data for fair usage. Otherwise, the training of higher values can outweigh the lower and the neural network will not train properly. In this dissertation the

data is normalized, as neither normalization showed better results than standardization of data even though neural networks performed much better with either presence than without one. Normalizing to mean (μ) 0 and standard deviation (σ) 1:

$$\mu = \left(\frac{\sum_i (x_i)}{N}\right), \tag{32}$$

$$\sigma = \left(\sqrt{\frac{\sum (x-\mu)^2}{N-1}}\right), \tag{33}$$

$$S_i = \left(\frac{S-i-\mu}{\sigma}\right). \tag{34}$$

The validation and test data must be also normalized using the statistics (μ and σ) that were computed from the training data.

As a neural network is created for each desired output there is no need to standardize the output. The output standardization is usually used if there are more than one output and they are in different order, hence affecting the way weights converge during the learning process.

The statistical data is then collected to calculate the root mean square error (RMSE) and R^2 values for both sets.

The created model may fit perfectly to the training data set, though it can still not qualify as a very good model to represent the output for the given process at other points. For that reason, the verification data set is created so that the points are at the different locations. If the verification data set RMSE and R^2 values do not differ much from the training values, then the model has trained correctly, otherwise it must have been overfitted or trained improperly.

If the neural network model did not train correctly, the training parameters of the model can be adjusted or additional sample points can be collected from the circuit simulation. In general, the neural networks is an accurate metamodel for the PLL circuit components.

6.4. Comparative Analysis of the Metamodels

Various components of the PLL including the ring oscillator has been optimized using a metamodels. The different sampling techniques are examined for metamodel generation to optimize the amount of samples needed to create one.

Two metamodels are generated for each circuit parameter, one for frequency and one for power. The resulting metamodels are chosen to be fitted into the polynomial progression form:

$$y = \sum_{i,j=0}^{k} (a_{ij} x_1^i x_2^j), \tag{35}$$

where y is the response frequency, $x = [W_n, W_p]$ is the vector of design variables and aij are the coefficients determined by the polynomial regression which we chose to be with $k = 4$.

The square root of mean square error (RMSE), shown in Equation 38, is used to compare the sampling data to the actual simulation conducted from parasitic netlist by using SPICE.

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{36}$$

$$RMSE = \sqrt{\frac{1}{N} SSE}, \tag{37}$$

$$= \sqrt{\frac{1}{N} \sum_{k=1}^{N} (y(x_k) - \hat{y}(x_k))^2}. \tag{38}$$

RMSE shows the departure of the metamodel from the true model, the simulation samples of the golden surface. RMSE is dependent on sum of squares error (SSE), which can be calculated as shown in Equation 36. The smaller the RMSE value, the better the metamodel [14]. All the sampling and calculations are done automatically using a simulator [6].

For the test case circuits, LHS of 100 samples to fit the metamodels, which was found earlier give very successful results. The calculated RMSE for these metamodels show that they have 99.7% accuracy for both power and frequency.

The parasitic-aware netlist is generated from the physical layout of Figure 3.9 and is parameterized. The physical layout is generated to take into consideration the parasitic effects of the circuit and made to be symmetrical. The symmetry of the layout provides the even-order distortion in the differential output waveform and up-conversion [26]. The wire width is maximized to minimize wire resistance and it also provided some space for future changing of the devices for after the optimization physical layout generation step.

CHAPTER 7

OPTIMIZATION ALGORITHMS FOR METAMODEL ASSISTED DESIGN

In this chapter different algorithms are presented that can be used to optimize mixed-signal design space exploration. In the fast metamodel-assisted design flow, once the metamodels are generated they are made available for use. The algorithms use these metamodels instead of actual circuits (netlists). The metamodels being mathematical functions (instead of physical descriptions, netlists) they work coherently with the optimization algorithms. The literature is rich in optimization algorithms and they can be explored in the current context. However, a selected number of algorithms are applied in the context of PLL components and metamodels.

7.1. The Simulated Annealing Algorithm

The term annealing comes from metallurgy. It is a technique that involves heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. Simulated annealing optimization is an extension of the Monte Carlo algorithm. The algorithm essentially simulates the annealing process of metals [36]. Therefore, the algorithm has a random component. Thus, two successive runs will produce different results. The steps of simulated annealing based search are presented in Algorithm 2.

Figure 7.1 shows the algorithm in action as it searches the constraint space. The starting point is chosen to be in the middle, and the step value is a random value which is normally distributed between 0 to $0.1 * W_{nmax}$ nm on the x-axis and 0 to $0.1 * W_{pmax}$ nm on the y-axis. The algorithm's temperature setting does not affect its performance for this particular circuit since it is very smooth, but it is usually set as a high number and need to be able to reach 0 when the inner loop completes its cycles. The max cycle time is set to 50 iterations for the inner loop with T = 100 degrees and according to the following expression:

$$Cooling\_Rate = (max\_iteration/T) - 1. \tag{39}$$

Algorithm 2 Simulated annealing algorithm for W_n and W_p.

Initialize iteration counter $Counter = 0$.
Initialize first feasible solution S_i=F(mid(W_n),mid(W_p)).
Determine initial Frequency for the solution S_i.
Initialize temperature T as T_i.
while (Frequency is varying) **do**
  $Counter$ =Maximum number of iterations.
  while ($Counter > 0$) **do**
    Generate random transition from S_i to S_i^*.
    if (S_i^* is acceptable solution) **then**
      $result = S_i^*$
      break both while loops.
    else
      Calculate change as: $\Delta Frequency = Frequency_S - Frequency_{S_i^*}$.
      if ($\Delta Frequency < 0$ random(0,1) $< e^{\frac{\Delta Frequency}{T}}$) **then**
        Update the solution with new solution: $S \leftarrow S_i^*$.
      end if
    end if
    $Counter = Counter - 1$.
  end while
  Decrease temperature as: $T = T \times Cooling\_Rate$.
end while
return $result$

For the current situation, those figures came out to be 0.5. From Table 7.1 it is observed that the algorithm reaches the first optimal solution in 10 iterations which is within 0.48% of the needed result within 0.77 ms.

Table 7.3 shows the results for LC-VCO multiobjective optimization which is done using two metamodels; one for power and one for frequency. The frequency metamodel is used as

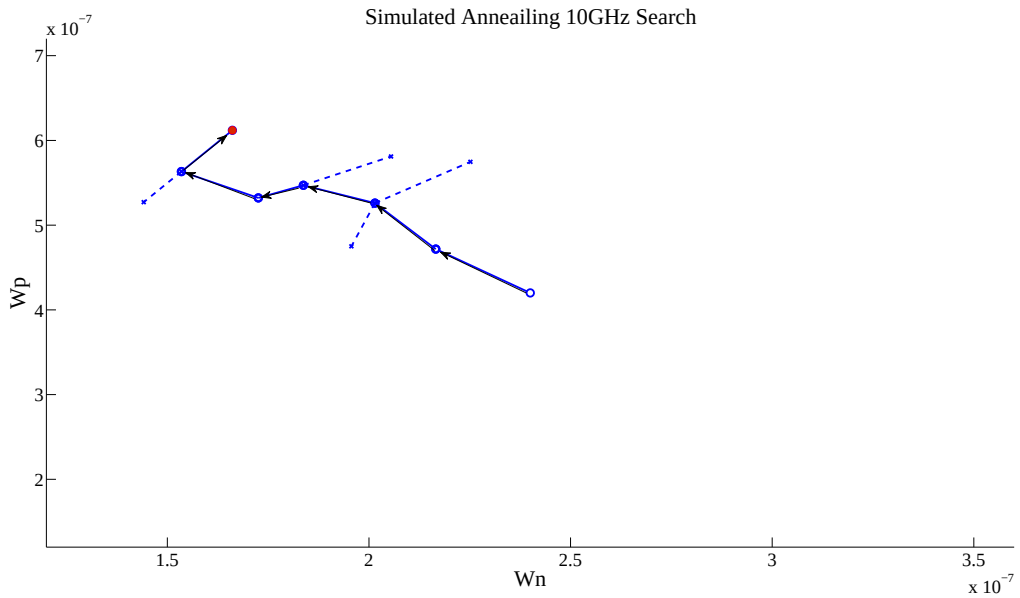


FIGURE 7.1. Simulated annealing for a 10 GHz objective. The search progressed in the direction of the arrow.

constraint in the optimization process and is set to 1% accuracy from 2.25 Ghz. The SAA algorithm is used to minimize the power with the given constraint. As a result the chosen two metamodels which were selected from the method above, are in the order of 4 and 5 for power and frequency respectfully. The accuracy of the metamodel prediction values are verified using the actual SPICE simulations and show that the metamodel error to actual simulation at the near optimal point is 0.047 mW for power and 0.02 Ghz for frequency. It is also an interesting find that the verification simulation showed much closer results to the target values with lower power and frequency being directly on target.

Table 7.4 shows the results of optimization conducted with the metamodel. The optimization results on the circuit itself, i.e. the netlist, are also presented. The metamodel generation time is generally in the order of seconds and depends on the maximum number of coefficients that can be present in the model, since the stepwise regression process is very much dependent on the number of coefficients. Higher order polynomials add more calculation time. The longest 6th order polynomial with 9 parameters took roughly 2 minutes to create. The optimization time is also in seconds in comparison to simulation optimization.

TABLE 7.1. Simulated annealing optimization for frequency of 45nm ring oscillator circuit with 5% bound.

Loop Iterations	Results Needed	Results Found	Accuracy	Time
Parasitic Netlist Optimization (Without Metamodel)				
35	9 GHz	8.97 GHz	0.33%	6.84 min
14	9.5 GHz	9.44 GHz	0.63%	2.73 min
15	10 GHz	10.07 GHz	0.31%	2.93 min
24	10.5 GHz	10.40 GHz	0.97%	4.69 min
16	11 GHz	10.96 GHz	0.36%	3.12 min
5	11.5 GHz	11.46 GHz	0.34%	0.98 min
3	12 GHz	11.99 GHz	0.08%	0.59 min
10	12.5 GHz	12.47 GHz	0.24%	1.95 min
Metamodeling Optimization				
32	9 GHz	8.96 GHz	0.48%	1.8 ms
18	9.5 GHz	9.41 GHz	0.94%	1.05 ms
10	10 GHz	10.05 GHz	0.48%	0.77 ms
19	10.5 GHz	10.40 GHz	0.96%	1.16 ms
13	11 GHz	10.95 GHz	0.49%	0.85 ms
4	11.5 GHz	11.48 GHz	0.22%	0.38 ms
2	12 GHz	11.98 GHz	0.16%	0.16 ms
12	12.5 GHz	12.42 GHz	0.63%	0.95 ms

Table 7.5 shows the final results for power consumption for all PLL components which was designed for an 180 nm CMOS process design kit (PDK). The initial power was measured at the lowest minimal values for each parameter considered in all circuits. The order of the metamodels are different and where picked from having the lowest RMSE values from Table 7.6. It is easy to observe that on average the metamodeling approach has reached better results than the simulated

TABLE 7.2. Simulated annealing optimization for frequency of the 180nm LC-VCO circuit for 1% bound.

Loop Iterations	Results Needed	Results Found	Accuracy	Time
Parasitic Netlist Optimization (Without Metamodel)				
34	2.3 GHz	2.31 GHz	0.44%	7.4 min
60	2.2 GHz	2.22 GHz	0.97%	15 min
24	2.1 GHz	2.08 GHz	0.99%	5.6 min
Metamodeling Optimization				
31	2.3 GHz	2.3 GHz	0.06%	23.8 ms
18	2.2 GHz	2.2 GHz	0.09%	18.1 ms
30	2.1 GHz	2.1 GHz	0.03%	25.0 ms

TABLE 7.3. Multiobjective optimization using power and frequency metamodels for 180nm LC-VCO circuit with 1% bound for 2.25Ghz and power minimization.

	Power	Frequency	Error to verification	Error to target
Prediction	0.153 mW	2.23 Ghz	0.047 mW	0.02 Ghz
Verification	0.110 mW	2.25 Ghz	–	0 Ghz
Polynomial order	4	5		

approach on actual circuit/netlist, even though simulated optimization has better results for divider. The final values for PLL component circuits after optimizations are shown in Table 7.6, Table 7.7, Table 7.8, and Table 7.9.

7.2. The Proposed Bee Colony Algorithm

Swarm intelligence artificial bee colony algorithm is used in this Section. For the purpose of comparison it is also integrated to be used in the local loop of the memetic algorithm. The

TABLE 7.4. Power metamodel versus SPICE optimization comparison.

Circuit (parameters)	Metamodel				Netlist	
	Order	Prediction (W)	Actual (W)	Time	Actual (W)	Time
Phase Detector (6)	2	6.66×10^{-9}	6.67×10^{-9}	0.17s	6.81×10^{-9}	$\approx$ 19 min
	3	6.50×10^{-9}	6.65×10^{-9}	0.42s		
	4	6.64×10^{-9}	6.71×10^{-9}	1.02s		
	5	6.22×10^{-9}	6.89×10^{-9}	2.26s		
	6	3.13×10^{-9}	6.77×10^{-9}	4.58s		
Charge Pump (4) (without data centering)	2	3.01×10^{-5}	3.10×10^{-5}	0.11s	3.11×10^{-5}	$\approx$ 18 min
	3	2.97×10^{-5}	3.12×10^{-5}	0.20s		
	4	3.01×10^{-5}	3.11×10^{-5}	0.35s		
	5	2.99×10^{-5}	3.12×10^{-5}	0.62s		
	6	3.12×10^{-5}	3.13×10^{-5}	1.02s		
Charge Pump (4) (with data centering)	2	7.41×10^{-6}	3.30×10^{-5}	0.11s	3.11×10^{-5}	$\approx$ 18 min
	3	7.19×10^{-6}	3.23×10^{-5}	0.21s		
	4	9.58×10^{-6}	3.96×10^{-5}	0.35s		
	5	2.53×10^{-5}	3.96×10^{-5}	0.61s		
	6	1.97×10^{-5}	4.08×10^{-5}	1.00s		
LC-VCO (2)	2	1.06×10^{-4}	7.11×10^{-5}	0.61s	9.54×10^{-5}	$\approx$ 18 min
	3	2.72×10^{-5}	7.10×10^{-5}	0.76s		
	4	4.56×10^{-5}	7.14×10^{-5}	0.99s		
	5	8.84×10^{-5}	7.14×10^{-5}	1.22s	9.54×10^{-5}	$\approx$ 18 min
	6	9.81×10^{-5}	7.14×10^{-5}	1.53s		
Divider (1)	2	5.02×10^{-5}	4.91×10^{-5}	0.3s	3.43×10^{-5}	$\approx$ 8 min
	3	1.20×10^{-5}	4.81×10^{-5}	1.05s	3.43 $\times 10^{--5}$	$\approx$ 8 min
	4	1.41×10^{-6}	5.32×10^{-5}	3.5s		
	5	2.25×10^{-4}	7.17×10^{-5}	11.25s		
	6	$\times 10^{-9}$	$\times 10^{-9}$	s		

TABLE 7.5. Final results of power optimization using metamodeling.

Components Components	Phase Detector	Charge Pump Pump	LC-VCO	Divider Divider	Total Power	Total Time
Initial Power	9.31 nW	21.84 μW	75.56 μW	60.57 μW	157.98 μW	
Metamodel Optimization	6.80 nW	3.10 μW	71.4 μW	48.1 μW	122.61 μW	7.48s
Simulation Optimization Optimization	6.87 nW	3.11 μW	95.4 μW	34.3 μW	132.82 μW	81 min

TABLE 7.6. Phase detector optimized values.

	W_{nDFF1}	W_{pDFF1}	W_{nDFF2}	W_{pDFF2}	W_n	W_p
Simulated Values	4.112e-7	4.304e-7	4.032e-7	4.224e-7	4.256e-7	4.336e-7
Metamodel Values	4.1837e-7	1.3833e-6	1.6294e-6	1.6453e-6	4.0293e-7	1.6606e-6

TABLE 7.7. The divider optimized values.

	M1	M2	M3	M4	M5	M6	M7	M8	M9
Simulated Values	4.69e-7	4.08e-7	4.20e-7	4.12e-7	4.81e-7	4.65e-7	4.45e-07	4.33e-7	4.66e-7
Metamodel Values	4.10e-7	4.37e-7	4.43e-7	4.03e-7	4.04e-7	4.71e-7	1.58e-6	1.13e-6	4.04e-7

TABLE 7.8. The LC-VCO optimized values.

	PM1 & PM2	NM1 & NM2
Simulated Values	3.189e-6	2.07e-5
Metamodel Values	3.0262e-6	6.1192e-6

general artificial bee colony algorithm (ABC) and their heuristics are thoroughly described in [50, 32, 31, 27].

TABLE 7.9. The charge pump optimized values.

	M1	M2	M3	M4
Simulated Values	4.128e-7	4.592e-7	5.168e-7	4.32e-7
Metamodel Values	4.0092e-7	4.2279e-7	1.072e-6	4.0966e-7

The Artificial Bee Colony (ABC) Optimization algorithm is based on the natural behavior of honey bees for finding best food source. The artificial bee colony divides bees into three categories: onlookers, scouts, and workers. The algorithm starts with bees being divided equally between onlookers and worker bees only. The initial random solution is assigned to worker bees. Worker bees search for food at the known random location. When bees return to hive the information is shared among the bees by performing the wiggly dance on the dance floor. The unemployed onlooker bees then choose the best food source and employ themselves to go search for more food around the area of the food source. The worker bees that the food source has been abandoned become scout bees and start searching for food randomly.

The workings of the meta-heuristic ABC algorithm used for optimization is described in algorithm 3. The proposed algorithm is a maximization algorithm and due to the random behavior of bees it can leave local maximum and potentially find the global maximum, given enough iterations. The FoM described in this algorithm can be based on target mixed-signal component. In this algorithm the boundary for each parameter is calculated using the following expression:

$$P = \{P.min + random(1) * P.max\} * buffer. \tag{40}$$

7.3. The Proposed Memetic Algorithm

Dawkin [10] has introduced the notion of a meme in 1976. In 1989 the term memetic algorithm (MA) was introduced in [43] by P. Moscato. The idea proposed a multilevel algorithm that combines evolutionary algorithms for global optimization with more powerful algorithms that are specifically used for local search. As the genetic algorithm approaches the global optimum, the local optimizer is applied to each offspring before it is inserted into the population. The unique

Algorithm 3 Artificial bee colony algorithm for multi-objective optimization.

Initialize maximum iterations = maxi and **Set** boundaries for each parameter P(i)= [min,max].
Define number of bees NumberBees. and **Initialize** the buffer value for worker bees disperse.
Initialize bee_matrix(3,NumberBees) = [workers, onlookers, scouts].
Set bee_matrix first half to be workers and other onlookers. and **Initialize** food sources.
for counter 1 to maxi **do**
    for i=1 to NumberBees **do**
        if bee_matrix(1,i)==1 **then**
            (1) **send** worker bee to a random known food source and **Calculate** FoM.
            if FoM is better than old **then**
                Update result and location.
            else
                Convert bee to onlooker.
            end if
        else
            if bee_matrix(1,i)==1 **then**
                (2) **send** onlooker bee and **Calculate** probability if the food source is good.
                if probability is high **then**
                    Send onlooker to random location for each P and calculate FoM.
                    if FoM is better than old **then**
                        Update result and location. and **Convert** bee to worker.
                    else
                        Convert bee to scout.
                    end if
                end if
            else
                (3) **send** scout bee, **Pick** the best result = best_r. and **Send** the scout to random location for each P.
                if FoM is better than old **then**
                    Update result and **Convert** bee to worker.
                end if
            end if
        end if
        if FoM is better than old **then**
            Update result and location.
        end if
    end for
end for
Return result and location.

aspect of the MA algorithm is that all chromosomes and offspring are allowed to gain some experience, through a local search, before being involved in the evolutionary process [43, 13].

There has been some discussion whether the memetic algorithm (MA) and genetic algorithm (GA) are identical. A genetic algorithm is a computational model that mimics the biological evolution, whereas a memetic algorithm, in contrast, mimics cultural evolution [10]. Even though the memetic algorithm (MA), has the same parameters that are involved in the algorithm as in genetic algorithms (GAs): population size, number of generations, crossover rate, and mutation rate in addition to a local search mechanism, it is not the case since memes can be thought of as units of information that are replicated while people exchange ideas. In an MA, a population consists solely of local optimum solutions [45]. The general description of different heuristics of memetic algorithms can be found in [45]. In [61], efficient MA-based materialized views selection algorithms are presented. Meta-Lamarckian learning is proposed for use in memetic algorithms in [44].

The proposed algorithm (Algorithm 4) is a heuristic memetic optimization algorithm. This includes the Artificial Bee Colony algorithm (Algorithm 3) at the local layer. As memes progress toward the goal on the global layer the local optimization does the fine tuning of the result for memes. Each artificial bee in the ABC algorithm acts as co-memes which are trying to find a better solution within bounds of the local search with the center as the main meme. The weight learning is used to adjust the probability of meme movement control on the global layer with the learning propagating from local to global layer. Since each meme, initially, has equal chance of moving in both directions the weights are used to teach memes to move in the appropriate direction. The weights are adjusted only when the meme reaches closer to optimal, since only then further direction can be estimated. The mutation of the meme is simulated by applying the random location to the meme instead of using the regular move with the learning weighs being reset. Essentially the ABC is the artificial representation of the bee colony behavior as bees try to find the best food source.

The function for FoM (Alg. 5) is designed to interface with either optimization algorithm. The FoM of interest is to minimize power as long as the PLL is working within the constraints.

Algorithm 4 Global Optimizer for Heuristic Memetic Algorithm.

Initialize initial population.

Initialize weights.

count = 0.

while count ¡ max_iterations **do**

count=count+1.

Evaluate all individual population.

for each individual in the population **do**

Select suitable memes.

Process with Alg. 2 for local improvements and

Replace selected meme with the improved solution.

Receive information of the improved location for that meme.

Adjust meme's weights.

end for

Calculate probability for random selection operations = prob.

if prob is crossover **then**

Swap selected meme's parameters.

else if prob is mutation **then**

Replace selected meme with random solution.

Reset selected meme's weights.

end if

end while

Return optimal value and location.

Multichannel multipoint distribution service (MMDS) standards are set in the FoM function (Alg. 5) for freq = 2.7 GHz $\pm$0.1% and l_time = 200 ns while Worldwide Interoperability for Microwave Access (WiMAX) freq = 2.5 GHz $\pm$0.1%. It can be seen that the initial design does not fit into these specifications. If the process to optimize this design to these specifications is done by hand it would take days, if not weeks to do so; adjusting the physical design multiple

Algorithm 5 FoM function for MMDS application

Receive P coordinates.
Calculate frequency from neural network with P parameters = freq.
if freq is within specification **then**
    Calculate locking time from neural network with P parameters = l_time.
    if l_time ¡ specifications **then**
        Calculate power from neural network with P parameters = power.
        FoM = 1/(power*1e3).
    else
        FoM = 0.
    end if
end if
Return FoM

times, correcting the W/L parameters, etc. Instead, the parameters that were chosen to describe the circuit are used to sample the design space using LHS design. The output is then recorded and further used to train the neural network with LHS design as input and each needed specifications for output. Another smaller set of LHS data is carefully created for the same design parameters, but is made sure that they are not the same as the training design set. This set will be used for verification of the neural network to make sure that the design is not over-fitted and is accurate.

Over-fitting is the phenomenon when the network gets worse instead of better after a certain point during training when it is trained to as low errors as possible. This is either because of long training or a large amount of neurons present in the hidden layer may make the network memorize the training patterns and stop adjusting the weights. There are several methods to avoid over-fitting. One method is regularization which tries to limit the complexity of the network such that it is unable to learn peculiarities. Another method is early stopping which aims to stop the training at the point of optimal generalization.

Both memetic and artificial bee colony algorithms are used to optimize the circuit to the above specifications. Since both algorithms are constructed to reach optimal maximum, the figure of merit is transposed to: FoM=1/(Power*1000) since the power was in mW.

7.4. Comparing the Memetic-Based Algorithm with Other Intelligent Algorithms

Table 7.10 shows the results for both algorithms running independently optimizing to reach better power minimization for MMDS constraints. Figure 7.2 compares the number of iteration for both algorithms. The memetic algorithm reaches more optimal result faster even though the ABC algorithm shows better results in the beginning. This is due to the ABC algorithm being purely stochastic, which does not enable the algorithm to always converge to the global maximum, even though it has a chance to do so with more iterations. The memetic algorithm starts off a little bit slow due to the learning process, but then reaches the result quite rapidly.

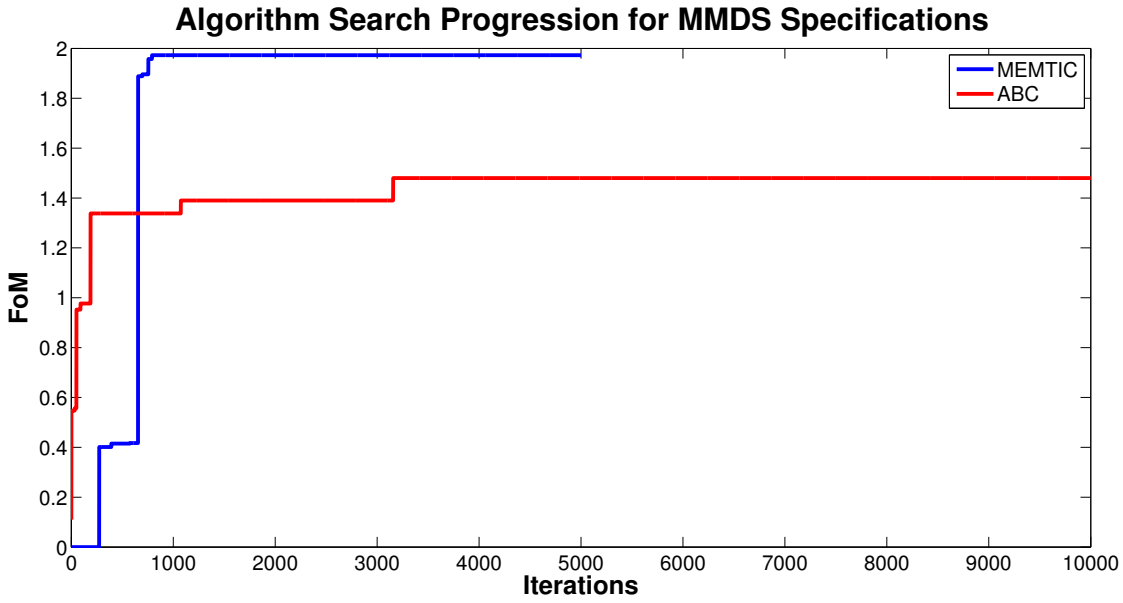


FIGURE 7.2. Comparison of artificial bee colony versus memetic based algorithm for MMDS specifications.

Table 7.11 shows the results for both algorithms running independently optimizing to reach better power minimization for MMDS constraints. Figure 7.3 shows the results as both algorithms are running independent optimizations trying to maximize FoM for WiMAX constraints. It can be observed that the Memetic algorithm reaches the optimum quite faster than the ABC algorithm.

TABLE 7.10. Parameter ranges and optimization results for MMDS specifications.

Circuit	Component	Parameter Name	min	max	MMDS	
					Memetic	ABC
Phase Detector	DFF1 PMOS	W_{ppd1}	0.4 μm	2 μm	0.77 μm	1.22 μm
	DFF1 NMOS	W_{npd1}	0.4 μm	2 μm	0.4 μm	0.61 μm
	DFF2 PMOS	W_{ppd2}	0.4 μm	2 μm	0.4 μm	0.68 μm
	DFF2 NMOS	W_{npd2}	0.4 μm	2 μm	0.4 μm	0.63 μm
	AND PMOS	W_{ppd3}	0.4 μm	2 μm	0.4 μm	1.64 μm
	AND NMOS	W_{npd3}	0.4 μm	2 μm	0.4 μm	1.01 μm
Charge Pump	M3, M4	W_{pCP1}	0.4 μm	2 μm	2.0 μm	2.77 μm
	M5, M6	W_{nCP1}	0.4 μm	2 μm	0.4 μm	1.67 μm
	M1, M2	W_{pCP2}	4 μm	20 μm	1.0 μm	1.6 μm
	M7, M8, M9	W_{nCP2}	2 μm	20 μm	0.77 μm	1.09 μm
LC-VCO	NM1, NM2	W_{nLC}	3 μm	20 μm	3.0 μm	6.7 μm
	PM1, PM2	W_{pLC}	6 μm	40 μm	13.9 μm	18.9 μm
Dividor	M5	W_{n1Div}	0.4 μm	2 μm	0.4 μm	0.43 μm
	M6	W_{n2Div}	0.4 μm	2 μm	0.4 μm	0.74 μm
	M7	W_{n3Div}	0.4 μm	2 μm	0.4 μm	0.41 μm
	M8	W_{n4Div}	0.4 μm	2 μm	0.4 μm	0.63 μm
	M9	W_{n5Div}	0.4 μm	2 μm	0.4 μm	0.7 μm
	M1	W_{p1Div}	0.4 μm	2 μm	0.77 μm	1.27 μm
	M2	W_{p2Div}	0.4 μm	2 μm	0.4 μm	1.17 μm
	M3	W_{p3Div}	0.4 μm	2 μm	0.77 μm	1.83 μm
	M4	W_{p4Div}	0.4 μm	2 μm	0.77 μm	1.65 μm

Table 7.12 shows the final solution outputs for both optimization algorithms. The memetic algorithm has shown better results for MMDS optimization by converging on much better solution than the bee-colony algorithm. The WiMAX specifications are very closely matched.

TABLE 7.11. Parameter ranges and optimization results for WiMAX specifications.

Circuit	Component	Parameter Name	min	max	WiMAX	
					Memetic	ABC
Phase Detector	DFF1 PMOS	W_{ppd1}	0.4 μm	2 μm	2.0 μm	2.0 μm
	DFF1 NMOS	W_{npd1}	0.4 μm	2 μm	1.7 μm	1.71 μm
	DFF2 PMOS	W_{ppd2}	0.4 μm	2 μm	1.25 μm	1.25 μm
	DFF2 NMOS	W_{npd2}	0.4 μm	2 μm	0.71 μm	0.7 μm
	AND PMOS	W_{ppd3}	0.4 μm	2 μm	1.8 μm	1.79 μm
	AND NMOS	W_{npd3}	0.4 μm	2 μm	0.56 μm	0.56 μm
Charge Pump	M3, M4	W_{pCP1}	0.4 μm	2 μm	1.72 μm	1.7 μm
	M5, M6	W_{nCP1}	0.4 μm	2 μm	0.43 μm	0.43 μm
	M1, M2	W_{pCP2}	4 μm	20 μm	1.81 μm	1.8 μm
	M7, M8, M9	W_{nCP2}	2 μm	20 μm	1.07 μm	1.07 μm
LC-VCO	NM1, NM2	W_{nLC}	3 μm	20 μm	19.45 μm	19.45 μm
	PM1, PM2	W_{pLC}	6 μm	40 μm	26.6 μm	26.6 μm
Dividor	M5	W_{n1Div}	0.4 μm	2 μm	0.88 μm	0.86 μm
	M6	W_{n2Div}	0.4 μm	2 μm	1.26 μm	1.26 μm
	M7	W_{n3Div}	0.4 μm	2 μm	1.72 μm	1.73 μm
	M8	W_{n4Div}	0.4 μm	2 μm	0.9 μm	0.92 μm
	M9	W_{n5Div}	0.4 μm	2 μm	0.98 μm	0.98 μm
	M1	W_{p1Div}	0.4 μm	2 μm	1.85 μm	1.85 μm
	M2	W_{p2Div}	0.4 μm	2 μm	1.83 μm	1.80 μm
	M3	W_{p3Div}	0.4 μm	2 μm	1.78 μm	1.74 μm
	M4	W_{p4Div}	0.4 μm	2 μm	2.0 μm	2.0 μm

As a final comparison, Table 7.13 shows the run time and convergence for both algorithms. The speedup was calculated for each in comparison to the Bee-Colony algorithm. Even though both of the algorithms complete the same amount of iterations, the speedup comes from the number

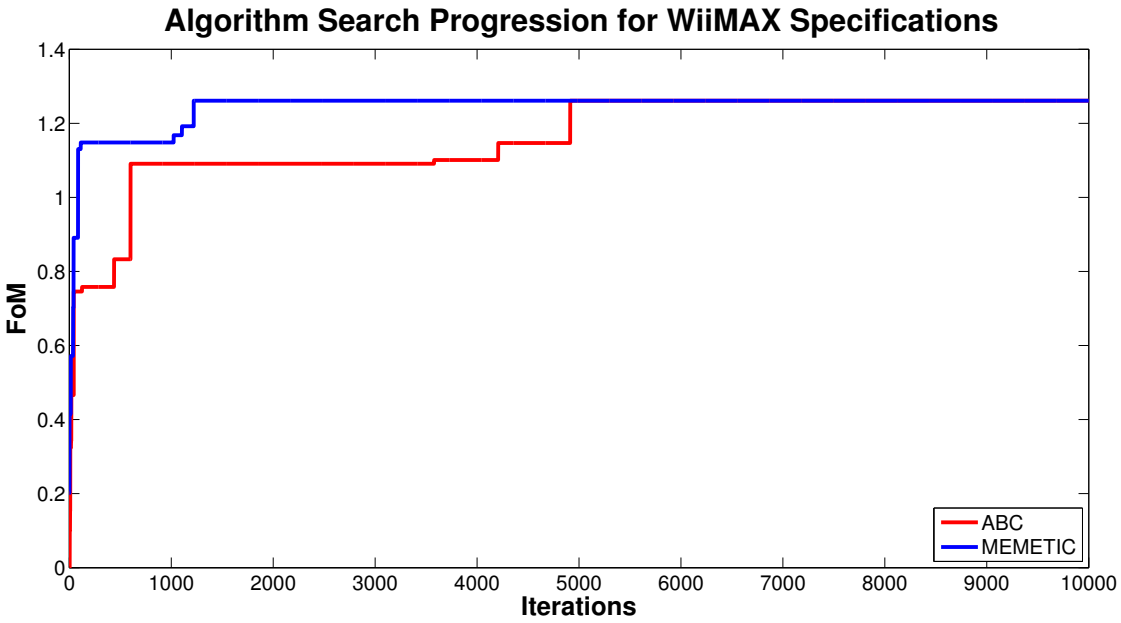


FIGURE 7.3. Comparison of artificial bee colony versus memetic based algorithm for WiMAX specifications.

TABLE 7.12. Final optimization results for different FoMs for WiMAX and MMDS PLLs.

	MMDS		WiMAX	
	Memetic	ABC	Memetic	ABC
Power	0.51 mW	0.68 mW	0.79 mW	0.79 mW
Locking Time	1.9 μs	1.58 μs	1.93 μs	1.92 μs
Frequency	2.702 GHz	2.703 GHz	2.502 GHz	2.502 GHz

of accesses to the metamodels, which Bee-Colony algorithm has to complete for every bee. As the memetic algorithm does not involve as many calculations at the global stage, it affects the calculation time.

TABLE 7.13. Comparison of algorithms for speed and convergence.

Algorithm	Simulation Time	Convergence (iter.)		Speedup
		WiMAX	MMDS	
ABC	≈12 min	4914	3193	1x
Memetic	≈5 min	1221	825	2.4x

CHAPTER 8

CONCLUSION AND FUTURE RESEARCH

In this dissertation, a novel design flow is presented for AMS-SoC component design using polynomial and non-polynomial metamodels. The commonly used sampling techniques are compared using a nano-CMOS ring oscillator in 45 nm and an LC-VCO in 180 nm technologies as case study circuits. The presented design flow can be used to speed up the design process of nanoscale circuits in general. The frequencies of the ring oscillator and LC-VCO were used as the objective functions for target specifications. It was shown that the parasitic effects play a major role in the figures-of-merit like power and frequency. A thorough analysis for various sampling data rates and methods demonstrates that uniform sampling techniques have better overall performance (in terms of accuracy) than the randomized and design of experiments (DOE) sampling techniques. Whether LHS or MLHS is more appropriate for a particular design depends on whether edge effects are important or not. It is important to note that during this research it is observed that sampling techniques and metamodeling techniques are technology independent. The LHS is typically preferable over MLHS because it covers the design space uniformly. At the same time, LHS provides more data for a small amount of randomness in the samples. Also LHS has a chance to cover the edges of the design space while MLHS does not.

For the second part of the study it is presented how to select a more accurate metamodel using the already sampled data. For the same two circuits, it was shown that higher order polynomials provide better overall fit and smaller mean error and error distribution. In addition, it is also shown that dual layer feed forward neural models have better accuracy than polynomial metamodels and can closely follow the behavior of the circuit and can be applied for global optimization. On the example of the PLL system, the models were generated from the physical layout netlist for 4 different characteristics of the circuit. If the designers need to optimize the circuit more than once or need to generate all the optimal solutions for the problem, the metamodeling approach comes out victor in saving design optimization time since the metamodels can be reused.

The optimization study has shown that metamodels can be used to simplify the design process for AMS-SoC circuits. With the use of the simulated annealing algorithm the study showed that metamodels can be used for optimization on AMS SoC circuits, such as RO and LC-VCO. On the example PLL, the circuit was parameterized with 21 parameters and optimized using the ABC optimization algorithm. The ABC algorithm is shown to be suitable for circuit optimization also and has performed very well. The final outcome of 90% power savings and average of 52% jitter minimization have been achieved with a minimal time of 100 simulations to generate the metamodel, in comparison to 100^21 simulations needed for an exhaustive search. The time savings are enormous ($\approx 10^{40}\times$ simulation time).

Following the bee-colony based optimization algorithm the search for more effective algorithms led the research to the memetic optimization algorithm. The proposed design flow using neural network models has been applied to an 180 nm PLL and the presented design approach was able to bring the circuit that was not within specifications to two different design specifications. The new heuristic memetic algorithm was presented and was shown to work considerably better than swarm based optimization algorithm (i.e. bee-colony based optimization). The memetic algorithm has converged on the near-optimal result 70% faster than bee-colony based optimization algorithm, while finding the same or even better solution. With the only time taxing portion being the metamodel creation process, which is model sampling, the 130 total simulations took approximately 12.5 hours, while optimization the process only took 15 minutes. This would be a considerable speedup over running the optimization on the SPICE netlist itself, or readjusting the layout manually.

For future research it would be interesting to see the limitations of the neural models on the amount of parameters they can handle with comparison to their accuracy and amount of samples that are needed to create them, as well as considering more complex architectures of neural models. In addition, further extensions to the memetic algorithm can be applied to improve its performance. Future research can also be conducted on process variation and temperature analysis.

BIBLIOGRAPHY

[1] Anuradha Agarwal, Glenn Wolfe, and Ranga Vemuri, *Accuracy Driven Performance macromodeling of Feasible Regions during Synthesis of Analog Circuits*, Proc. Great Lakes Symp. VLSI, GLSVLSI, 2005, pp. 482–487.

[2] M.A. Aguirre, J. Chavez, A. Torralba, and L.G. Franquelo, *Analog Design Optimization by Means of a Tabu Search Approach*, 1994 IEEE International Symposium on Circuits and Systems, 1994. ISCAS '94., vol. 1, 30 1994, pp. 375–378.

[3] Shubhankar Basu, Balaji Kommineni, and Ranga Vemuri, *Variation-Aware Macromodeling and Synthesis of Analog Circuits Using Spline Center and Range Method and Dynamically Reduced Design Space*, Proceedings of the 22nd International Conference on VLSI Design, 2009, pp. 433–438.

[4] K. Binder, *Monte-carlo methods in mathematical tools for physicists (ed g. l. trigg)*, Mathematical Tools for Physicists (ed G. L. Trigg), 2006.

[5] T. Binder, C. Heitzinger, and S. Selberherr, *A Study on Global and Local Optimization Techniques for TCAD Analysis Tasks*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 23 (2004), no. 6, 814–822.

[6] C.D.S. Inc., *Spectre Circuit Simulator User's Guide*, 2005.

[7] H.H.Y. Chan and Z. Zilic, *Modeling Layout Effects for Sensitivity-based Analog Circuit Optimization*, Quality of Electronic Design, 2005. ISQED 2005. Sixth International Symposium on, March 2005, pp. 390 – 395.

[8] C.-Y. Chao and L. Milor, *Performance Modeling of Analog Circuits Using Additive Regression Splines*, Proceedings of the IEEE 1994 Custom Integrated Circuits Conference, 1994., May 1994, pp. 301–304.

[9] N. Damavandi and S. Safavi-Naeini, *A Hybrid Evolutionary Programming Method for Circuit Optimization*, IEEE Transactions on Circuits and Systems I: Regular Papers 52 (2005), no. 5, 902–910.

[10] R. Dawkin, *The Selfish Gene*, Oxford, U.K.: Oxford Univ. Press, 1976.

[11] Y. Delican, R. A. Vural, and T. Yildirim, *Artificial Bee Colony Optimization based CMOS Inverter Design Considering Propagation Delays*, Proceedings of the XIth International Workshop on Symbolic and Numerical Methods, Modeling and Applications to Circuit Design (SM2ACD), 2010, pp. 1–5.

[12] M. Ding and R. Vemuri, *Efficient Analog Performance Macromodeling Via Sequential Design Space Decomposition*, Proceedings of the 19th International Conference on VLSI Design, January 2006, pp. 553–556.

[13] E. Elbeltagi, T. Hegazy, and D. Grierson, *Comparison Among Five Evolutionary-based Optimization Algorithms*, Advanced Engineering Informatics 19 (2005), 43–53.

[14] Kai-Tai Fan, Runze Li, and Agus Sudjianto, *Design and Modeling for Computer Experiments*, Chapman and Hall/CRC, London, 2006.

[15] Pradeep Fernando and Srinivas Katkoori, *An Elitist Non-Dominated Sorting Based Genetic Algorithm for Simultaneous Area and Wirelength Minimization in VLSI Floorplanning*, Proceedings of the 21st International Conference on VLSI Design, 2008, pp. 337–342.

[16] F. Gardner, *Charge-Pump Phase-Lock Loops*, Communications, IEEE Transactions on [legacy, pre-1988] 28 (1980), no. 11, 1849–1858.

[17] 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 (2012), no. 1, 26–36.

[18] O. Garitselov, S. P. Mohanty, E. Kougianos, and P. Patra, *Bee Colony Inspired Metamodeling Based Fast Optimization of a Nano-CMOS PLL*, Proceedings of the 2nd IEEE International Symposium on Electronic System Design (ISED), 2011.

[19] O. Garitselov, S.P. Mohanty, E. Kougianos, and P. Patra, *Nano-CMOS Mixed-Signal Circuit Metamodeling Techniques: A Comparative Study*, 2010 International Symposium on Electronic System Design (ISED), 2010, pp. 191–196.

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

[21] Dhruva Ghai, Saraju P. Mohanty, and Elias Kougianos, *Parasitic Aware Process Variation Tolerant Voltage Controlled Oscillator (VCO) Design*, Proceedings of the 9th International Symposium on Quality of Electronic Design, 2008, pp. 330–333.

[22] ———, *A process and supply variation tolerant nano-cmos low voltage, high speed, a/d converter for system-on-chip*, Proceedings of the 18th ACM Great Lakes Symposium on VLSI, 2008, pp. 47–52.

[23] ———, *Design of Parasitic and Process-Variation Aware Nano-CMOS RF Circuits: A VCO Case Study*, IEEE Trans. VLSI Syst. 17 (2009), no. 9, 1339–1342.

[24] D. Gorissen, L. De Tommasi, W. Hendrickx, J. Croon, and T. Dhaene, *RF Circuit Block Modeling via Kriging Surrogates*, Microwaves, Radar and Wireless Communications, 2008. MIKON 2008. 17th International Conference on, May 2008, pp. 1–4.

[25] S.C. Gupta, *Phase-Locked Loops*, Proceedings of the IEEE 63 (1975), no. 2, 291–306.

[26] 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*, Proc. Eur., Microw. Conf., Oct 2005, pp. 31–34.

[27] R. Hedayatzadeh, B. Hasanizadeh, R. Akbari, and K. Ziarati, *A Multi-Objective Artificial Bee Colony for Optimizing Multi-Objective Problems*, 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 5, August 2010, pp. 277–281.

[28] S. Huband, P. Hingston, L. Barone, and L. While, *A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit*, IEEE Transactions on Evolutionary Computation 10 (2006), no. 5, 477–506.

[29] R. L. Iman, *Latin hypercube sampling*, Encyclopedia of Quantitative Risk Analysis and Assessment, 2008.

[30] S. Kang and Y. Leblebici, *CMOS Digital Inegrated Circuits*, 3rd ed., McGraw Hill, New York, 2003.

[31] D. Karaboga and B. Basturk, *On the Performance of Artificial Bee Colony (ABC) Algorithm*, Applied Soft Computing 8 (2008), 687–697.

[32] D. Karabora and B. Akay, *A Comparative Study of Artificial Bee Colony Algorithm*, Applied Mathematics and Computation a (2009), no. 214, 108–132.

[33] Elias Kougianos and Saraju P. Mohanty, *Impact of gate-oxide tunneling on mixed-signal design and simulation of a nano-cmos vco*, Microelectronics Journal 40 (2009), no. 1, 95–103.

[34] Chin-Cheng Kuo, Meng-Jung Lee, Chien-Nan Liu, and Ching-Ji Huang, *Fast Statistical Analysis of Process Variation Effects Using Accurate PLL Behavioral Models*, IEEE Transactions on Circuits and Systems I: Regular Papers 56 (2009), no. 6, 1160–1172.

[35] William Liu, *Mosfet Models for Spice Simulation, Including BSIM3v3 and BSIM4*, New York : Wiley-Interscience Publication, c2001.

[36] N. Metropolis, A. Rosenbluth, A. Teller, and E. Teller, *Equation of State Calculations by Fast Computing Machines*, Journal of Chemical Physics 21 (1953), 1087.

[37] S. P. Mohanty, N. Ranganathan, E. Kougianos, and P. Patra, *Low-Power High-Level Synthesis for Nanoscale CMOS Circuits*, 1st ed., Springer, 2008.

[38] Saraju P. Mohanty, *A secure digital camera architecture for integrated real-time digital rights management*, Journal of Systems Architecture - Embedded Systems Design 55 (2009), no. 10-12, 468–480.

[39] ———, *Unified Challenges in Nano-CMOS High-Level Synthesis*, Proceedings of the 22nd International Conference on VLSI Design, 2009, pp. 531–531.

[40] Saraju P. Mohanty and Dhiraj K. Pradhan, *Uls: A dual-v_{th}/high-kappa nano-cmos universal level shifter for system-level power management*, JETC 6 (2010), no. 2.

[41] Mohanty, S.P. and Ghai, D. and Kougianos, E., *A P4VT (Power Performance Process Parasitic Voltage Temperature) Aware Dual-VTh Nano-CMOS VCO*, VLSI Design, 2010. VLSID '10. 23rd International Conference on, 2010, pp. 99–104.

[42] Gerardo Montemayor-Garcia and Gregorio Toscano-Pulido, *A Study of Surrogate Models for Their use in Multiobjective Evolutionary Algorithms*, 8th International Conference on

Electrical Engineering Computing Science and Automatic Control (CCE), 2011, Oct. 2011, pp. 1–6.

[43] P. Moscato, *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Toward Memetic Algorithms*, Technical Report Caltech Concurrent Computation Program. Pasadena:California Institute of Technology, 1989.

[44] Yew Soon Ong and A.J. Keane, *Meta-Lamarckian Learning in Memetic Algorithms*, IEEE Transactions on Evolutionary Computation. 8 (2004), no. 2, 99–110.

[45] Yew-Soon Ong, Meng-Hiot Lim, Ning Zhu, and Kok-Wai Wong, *Classification of Adaptive Memetic Algorithms: A Comparative Study*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 36 (2006), no. 1, 141–152.

[46] J. Park, K. Choi, and D. J. Allstot, *Parasitic-Aware Design and Optimization of a Fully Integrated CMOS Wideband Amplifier*, Proceedings of the Asia South Pacific Design Automation Conference, 2003, pp. 904–907.

[47] R.R.B. Sheen, O.T.C. Chen, and R.C.H. Chang, *A 1.3 V 1.04 GHz-1.30 GHz CMOS Phase-Locked Loop*, Proceedings of the 40th Midwest Symposium on Circuits and Systems, 1997, 1997, pp. 569–572.

[48] J.F. Swidzinski, D. Alexander, M. Qu, and M.A. Styblinski, *A Systematic Approach to Statistical Simulation of Complex Analog Integrated Circuits*, 1997 2nd International Workshop on Statistical Metrology, June 1997, pp. 86–89.

[49] K. Takemura, T. Koide, H.J. Mattausch, and T. Tsuji, *Analog-circuit-component optimization with genetic algorithm*, The 2004 47th Midwest Symposium on Circuits and Systems, 2004. MWSCAS '04., vol. 1, July 2004, pp. 489–492.

[50] Dusan Teodorovic, Panta Lucic, Goran Markovic, and Mauro Dell' Orco, *Bee Colony Optimization: Principles and Applications*, 8th Seminar on Neural Network Applications in Electrical Engineering, 2006. NEUREL 2006., September 2006, pp. 151–156.

[51] Rajesh Amratlal Thakker, Maryam Shojaei Baghini, and Mahesh B. Patil, *Low-Power Low-Voltage Analog Circuit Design Using Hierarchical Particle Swarm Optimization*, Proceedings of the 22nd International Conference on VLSI Design, 2009, pp. 427–432.

[52] W.C.M. van Beers and J.P.C. Kleijnen, *Kriging Interpolation in Simulation: a Survey*, Proceedings of the 2004 Winter Simulation Conference, 2004., vol. 1, December 2004.

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

[54] Zuoding Wang, *An Analysis of Charge-Pump Phase-Locked Loops*, IEEE Transactions on Circuit and Systems-I: Fundamental Theory and Applications 52 (2005), no. 10, 2128–2138.

[55] G. Wolfe and R. Vemuri, *Extraction and Use of Neural Network Models in Automated Synthesis of Operational Amplifiers*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 22 (2003), no. 2, 198–212.

[56] ———, *Extraction and Use of Neural Network Models in Automated Synthesis of Operational Amplifiers*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 22 (2003), no. 2, 198–212.

[57] Likun Xia, Ian M. Bell, and Antony J. Wilkinson, *A Robust Approach for Automated Model Generation*, Proceedings of the 4th International Conference on Design Technology of Integrated Systems in Nanoscal Era, 2009, pp. 281–286.

[58] Jianjun Xu, M. C. E. Yagoub, Runtao Ding, and Q. J. Zhang, *Feedforward Dynamic Neural Network Technique for Modeling and Design of Nonlinear Telecommunication Circuits and Systems*, Proceedings of the International Joint Conference on Neural Networks, 2003, pp. 930–935.

[59] Guo Yu and Peng Li, *Yield-aware analog integrated circuit optimization using geostatistics motivated performance modeling*, IEEE/ACM International Conference on Computer-Aided Design, 2007. ICCAD 2007., Nov. 2007, pp. 464–469.

[60] Q. J. Zhang and K. C. Gupta, *Neural networks for rf and microwave design*, 1st ed., Artech House, Inc., Norwood, MA, USA, 2000.

[61] Qingzhou Zhang, Xia Sun, and Ziqiang Wang, *An Efficient MA-Based Materialized Views Selection Algorithm*, IITA International Conference on Control, Automation and Systems Engineering, 2009. CASE 2009., July 2009, pp. 315–318.

[62] Ying Zhao, Jun Gao, and Xuezhi Yang, *A Survey of Neural Network Ensembles*, International Conference on Neural Networks and Brain, 2005. ICNN B '05., vol. 1, October 2005, pp. 438–442.