Accurate Polynomial Metamodeling-Based Ultra-Fast Bee Colony Optimization of a Nano-CMOS PLL

Oleg Garitselov[1], Saraju P. Mohanty[2], and Elias Kougianos[3]
Nano-Systems Design Laboratory (http://nsdl.cse.unt.edu)[1,2,3]
Department of Computer Science and Engineering[1,2]
Department of Engineering Technology[3]
University of North Texas, Denton, USA.[1,2,3]
Email-ID: omg0006@unt.edu[1], saraju.mohanty@unt.edu[2], and eliask@unt.edu[3].

Abstract

Significant increase in design cycle time is caused by the design and optimization complexity of Analog/Mixed-Signal System-on-a-Chip (AMS-SoC) components as the technology moves deeper into the nanoscale domain. In this paper, a two-tier design methodology is proposed that greatly reduces design cycle time by combining accurate polynomial metamodeling and intelligent optimization. In this methodology, the parasitic-aware netlist description of an AMS-SoC component is converted into an accurate metamodel (a mathematical function or algorithm) which minimizes the time for design space exploration. Bee Colony Optimization (BCO) is subsequently used for optimization of the nano-CMOS AMS circuit. Five distinct metamodels with 21 parameters each are created for corresponding Figures of Merit (FoMs) to perform AMS-SoC component design space exploration. For a specific case study, a 180nm LC Voltage Controlled Oscillator (LC-VCO) based Phase-Locked Loop (PLL) frequency generator circuit is used this paper. The proposed optimization achieved approximately 90% power and 52% jitter reduction in comparison to the baseline design while maintaining the locking time of the PLL system. In comparison to an exhaustive simulation based design optimization approach, the proposed design flow can be up to 10^{20} times faster and hence has potential to greatly reduce design effort and chip cost.

Index Terms

Polynomial Metamodeling, Design Space Sampling, Phase-Locked Loop (PLL), Power Dissipation, Bee Colony Algorithm, Design Optimization

I. Introduction

The physical design (layout) stage of Analog/Mixed-Signal (AMS) circuits and systems is quite complex in nature and very time consuming as it is a manual process. It is hard to accurately predict the output of these AMS circuits and systems due to numerous parasitic elements involved when they are realized at the physical design level of the chip [1], [2]. The parasitic elements complicate circuit simulation and make traditional, continuous time numerical methods inefficient. Therefore sophisticated tools are used to simulate such Analog/Mixed-Signal System-on-a-Chip (AMS-SoC) components. However, for larger systems the simulation complexity results in run times of days to weeks for a single design space point. Thus, there is a pressing need for design methodologies that provide: (1) Fast simulation for characterization and verification of nano-CMOS AMS systems and circuits. (2) Fast layout optimization of complex nano-CMOS AMS systems. (3) Fast design space exploration and optimization convergence to reduce design cycle time under current strict time-to-market constraints.

Design optimization using the actual circuit (i.e. SPICE netlist description) is a pressing issue and has been addressed in the current literature. The existing approaches speedup the design cycle in various ways including fast algorithms, fast design sampling, or simplified circuit models (also known as *macromodels*). A macromodel, which is a simplified model of the actual circuit, is hard to create and keeps the design verification and optimization cycle

Corresponding Author: **Elias Kougianos**, University of North Texas, Department of Engineering Technology, 3940 N. Elm St., Suite F115, Denton, TX 76207-7102, USA, Office : +1 (940) 891-6708, Fax : +1 (940) 565-2666, Email : eliask@unt.edu.

closely coupled to analog simulation tools. Several current research works consider macromodeling to reduce the complexity of the circuit and system under design [3], [4], [5], [6]. However, the design optimization when the full-blown parasitic-aware SPICE netlist is used remains a time-consuming and resource-intensive task.

The design flow presented in this paper relies instead on *metamodels* which are predictive mathematical expressions or algorithms (also called surrogated models) and decouples the design and optimization from the actual analog circuit simulator [7], [8], [9]. A metamodel is essentially a predictive mathematical formula or process created on statistical data for a given figure of merit (FoM) such as power, frequency, jitter, leakage, and phase noise. If the design of the circuit requires more than one concurrent FoM optimization (possibly constrained), multiple metamodels, one for each FoM can be generated from the same sample of the design space. Metamodels provide enormous time savings during the optimization phase since the optimization is done on a simple formula instead of running one or more complex analog simulations for each optimization iteration. Metamodels are language and tool independent allowing the designer to use more powerful tools for the optimization and verification stages of the design. Metamodels are also reusable, which makes them a powerful IP, saving time for designers who create multiple designs with similar specifications. The same metamodels can also be used for ultra-fast process variation analysis. Other engineering and scientific fields use metamodeling especially on very costly or time consuming product designs [10].

The rest of the paper is organized in the following manner: Section II highlights the contributions of this paper. Section III discusses prior related research. The PLL circuit used as a case study is discussed in Section IV. Section V describes the metamodel-based design flow along with the metamodel generation technique and bee colony optimization algorithm. The optimization results are shown in Section VI. The paper is concluded with directions for future research in Section VII.

II. Contributions of this Paper

Addressing the design complexity and long design cycle time at the physical design level has been a challenging research problem; in particular, when full-blown (RCLK) parasitics are extracted. The current paper demonstrates how to adjust traditional design flows to significantly reduce the time and complexity of the entire process. The novel contributions of this paper to the state of the art are as follows:

- The paper presents a new design flow that applies layout-accurate polynomial metamodels, intelligent bee colony optimization algorithm, and single-manual-layout iteration for ultra-fast AMS-SoC component design exploration.
- The paper addresses how to create accurate, parasitic-aware metamodels for complex nanoscale AMS circuits and system using a phase-locked loop (PLL) as a case study. Accurate polynomial metamodels are created for 21 design parameters from only 100 SPICE-level simulations.
- AMS-SoC component circuit optimization is conducted using the bee colony optimization algorithm for the first time. The effectiveness of the proposed approach is demonstrated by conducting multi-objective optimization and reaching the desired specifications for the desired physical design of the PLL. The optimization is performed exclusively on the metamodels and its efficiency is proven by modifying the baseline design according to the results of the metamodel analysis and demonstrating on-spec operation of the circuit.
- The proposed design flow which uses the bee optimization algorithm is shown to perform orders of magnitude faster than analog simulator based approaches.

III. Related Prior Research

The current paper simultaneously utilizes layout-accurate metamodels and an intelligent bee-colony optimization algorithm in the context of a single manual-iteration design flow. These three aspects ensure ultra-fast design exploration and to the authors' best knowledge this is the first time this approach has been proposed in the literature. Thus, VLSI metamodeling research and AMS circuit fast optimization are considered as prior related literature.

Many function based approaches have been investigated to simplify or predict circuit behavior using predictive functions that are derived from statistical analysis and mathematical algorithms. In [8], different sampling techniques are examined for generating polynomial metamodels for small circuit optimization. In [11], support vector machine (SVM)-based machine learning is proposed as a surrogate for expensive circuit-level simulation. Metamodeling is used in [12] for creating an inductor for CMOS circuits using mathematical formulas for the model. A VCO

parametric metamodeling approach is given in [13]. A statistical wire-length estimation approach using surrogate modeling is proposed in [14]. Posynomial modeling for gate sizing is done in [15]. The author is using Response Surface Methodology (RSM) to approximate process variation of a low noise amplifier in [16]. These research works present metamodeling techniques, but do not provide a complete design optimization flow, as in the current paper.

Intelligent algorithms have also been explored for optimization of circuits. Intelligent optimization approaches including genetic algorithm [17], particle swarm optimization algorithms [18], or artificial bee colony [19] have been explored for circuit optimization. Relevant research that apply bee colony optimization in a VLSI context are now discussed. In [19], the artificial bee colony optimization algorithm is investigated considering the transient performance of a CMOS inverter circuit. In [20], a CMOS Miller operational transconductance amplifier is optimized using artificial bee colony algorithm. In [21], the bee colony algorithm is used to determine the different parameters of a Schottky barrier diode. Thus, in the current literature, the artificial bee colony optimization algorithms iterate over very small circuits and hence their scalability to large circuits is not feasible.

IV. Phase Locked Loop (PLL) Circuit Design

The Phase Locked Loop (PLL) is a widely used component in most AMS Systems-on-a-Chip (AMS-SoC), processors, Field-Programmable-Gate Arrays (FPGAs) etc. PLL frequency generators have two functions: indirect frequency multiplication or frequency demodulation. The PLL, as shown in Fig. 1, is a single closed loop feedback control system that keeps the generated higher clock signal (Clock Out) in close phase relationship to the reference signal (Clock In). The PLL system has several components: phase detector, charge pump, loop filter, LC-VCO, and frequency divider. The phase frequency detector produces two output signals UP and DOWN which are proportional to the difference of the output signal phase and input clock signal phase. The output signals of the phase detector drive the charge pump which provides the current to the second-order passive loop filter. The output signal from the filter becomes the control voltage for the VCO which oscillates according to the control voltage input. The frequency of the oscillator is typically different from the PLL input frequency. The frequency of the output signal from the LC-VCO is the output (Clock Out) signal of the PLL. This signal is looped back to the divider, which divides the frequency by some integer, to recover the reference frequency range. The easiest implementation for the divider is to keep the divider ratio a power of two since they are straightforward to design. When the PLL reaches the locked state, the frequency of the output should be an exact multiple of the input frequency [22]. During acquisition of the lock stage, the PLL responds to the frequency and therefore the phase of the input clock signal and tries to adjust itself until it becomes in phase (locked) with the input signal. Therefore it is essential to have the input signal as jitter free as possible, which is usually accommodated by off chip crystal oscillators which have low frequency ranges.

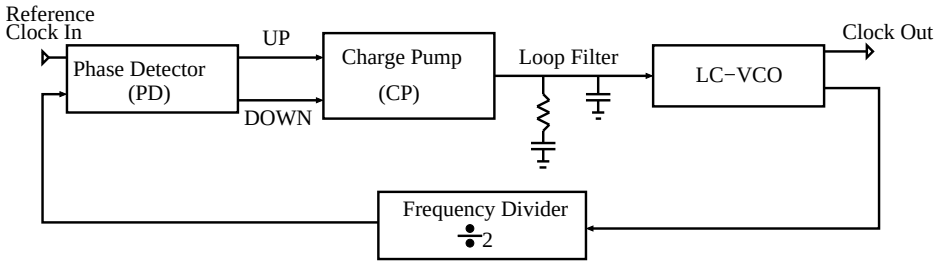
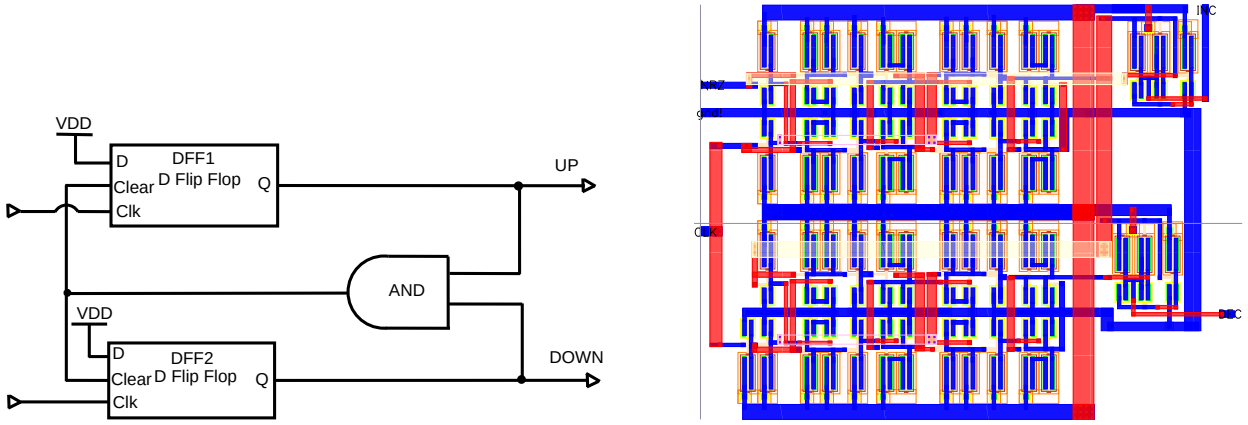


Fig. 1. Block diagram of a PLL.

A. Phase Detector

The phase detector is an analog mixer or an asynchronous sequential logic circuit used to detect mismatch between phase or frequency between two signals [23]. Two output signals of the phase detector (UP and DOWN) are active when the system is out of lock and produces pulses directly proportional to the phase difference of the two clock signals. These signals are then used to direct the charge pump to supply charge in proportion to the phase error detected. The logical design of the phase detector using two D flip-flops and one AND gate is shown in Fig. 2(a). The physical layout realization of the phase detector is shown in Fig. 2(b).



(a) High-level representation of the phase detector.

(b) Physical design of the phase detector circuit.

Fig. 2. Design of the phase detector circuit.

B. Charge Pump and Loop Filter

The charge pump and loop filter are used to stabilize spurious fluctuations of current and switching time so that the spurs in the VCO input are minimum. An efficient charge pump circuit provides constant current to the loop filter which integrates this current from the UP and DOWN signals received from the phase detector and the result is a continuously changing control voltage applied to the VCO. The combination of the two circuits creates analog signals for controlling the locked oscillator from the digital signal provided by the phase detector [24]. The circuit diagrams of the charge pump and loop filter are shown in Fig. 3(a) and Fig. 3(b), respectively.

The loop filter determines the dynamic characteristics of the PLL and affects tracking and capture ranges, and maximum slew rate. A low-pass RC filter is designed to pass frequency signals within the range of the VCO to smooth the voltage spurs coming from the charge pump and slow down VCO frequency transitions. The component values for the filter are considered in such a way that the cutoff frequency of the filter is approximately equal to the maximum frequency of the LC-VCO so that it rejects signals at frequencies above its maximum frequency. The loop filter is designed to match the specifications required by the application. The physical design of the charge pump and loop filter is shown in Fig. 3(c).

C. LC Voltage Controlled Oscillator

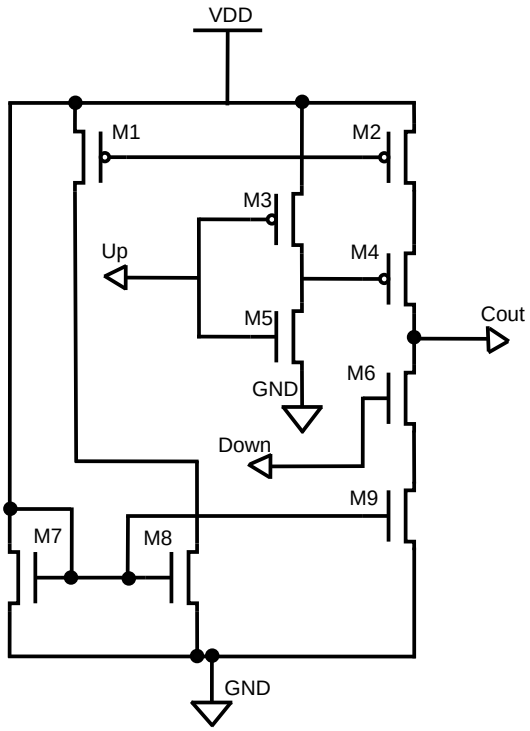
The LC-VCO is an oscillator that controls the target frequency of the PLL. It includes an LC tank and is controlled by 2 symmetrical voltage controlled capacitors (varactors) C_1 and C_2 that adjust the capacitance of the LC tank to respond to the voltage inputs. The LC-VCO circuit design is shown in Fig. 4(a). The designed LC-VCO is targeted towards a 2.6 GHz frequency with approximately 0.5 GHz tuning range. The physical layout realization of the LC-VCO is shown in Fig. 4(b).

D. Frequency Divider

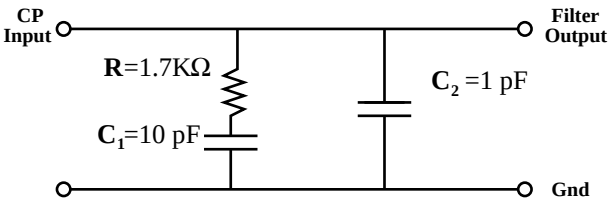
The frequency divider shown in Fig. 5(a) is implemented using true single phase clock logic. When a continuous train of pulse waveforms at fixed frequency is fed to it as an input signal, an output signal of approximately half the frequency of the input signal can be obtained. If used in a cascaded fashion it can also be used to provide division by 4, 8, etc. The circuit is comparable to a two-phase static D flip-flop operation. The physical layout of the divide by 2 circuit is shown in Fig. 5(b).

E. PLL Circuit Characterization

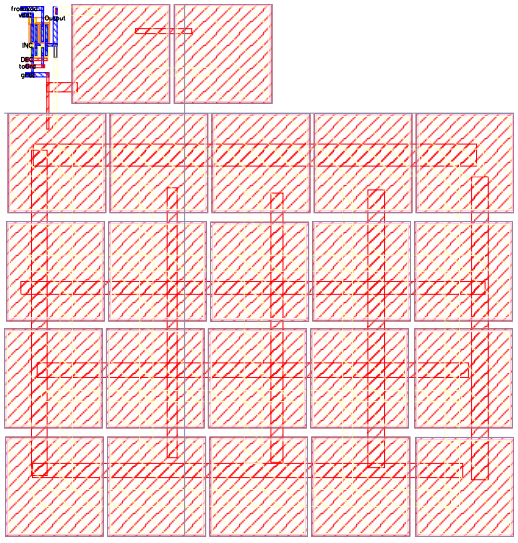
The logical and physical design of the PLL was performed in a 180 nm CMOS technology with a target frequency of 2.6 GHz. The characterization of the target FoMs is presented in Table II. The initial design showed acceptable phase noise characteristics with -172 dBc/Hz @ 1 MHz offset. The PLL circuit is characterized for output frequency, power, vertical and horizontal jitter (to simplify the phase noise calculations), and locking time which is used for



(a) Schematic of the charge pump circuit.



(b) Schematic of the loop filter circuit.



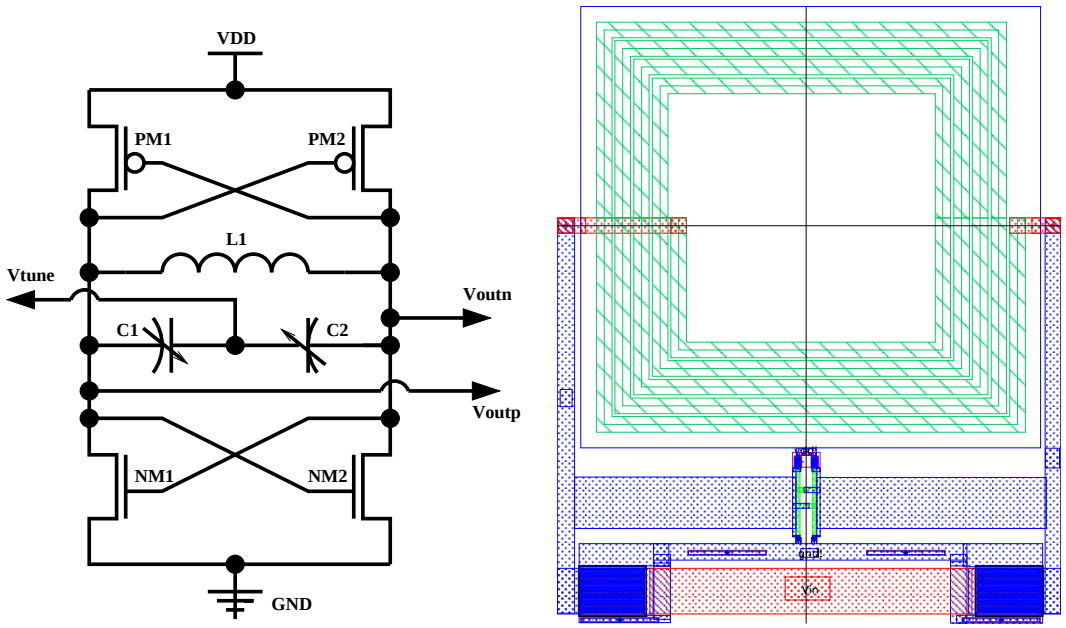
(c) Physical design of the charge pump and RC filter.

Fig. 3. Design of the charge pump and loop filter circuits.

correctness of operation. A separate metamodel is created for each FoM from the same sample of the design space. Each single transient simulation calculates all FoMs so the number of simulations required does not depend on the number of metamodels than need to be generated.

V. Proposed Accurate Polynomial Metamodel Based Bee Colony Optimization of PLL

This section introduces the proposed ultra-fast and yet accurate design flow that has 3 unique features to accomplish this goal. It is a flow that uses single manual layout iteration with two layout stages at most: one initial and one optimal. It uses layout-accurate polynomial metamodels which are generated from a parasitic-aware (RCLK - Resistance, Capacitance, Self and Mutual inductance)) netlist of the PLL circuit. The optimization is then performed using the polynomial metamodels and not the actual circuit netlist, thus significantly speeding up the



(a) Schematic diagram of the LC-VCO. (b) Physical layout of the LC-VCO.

Fig. 4. Design of LC-VCO circuit.

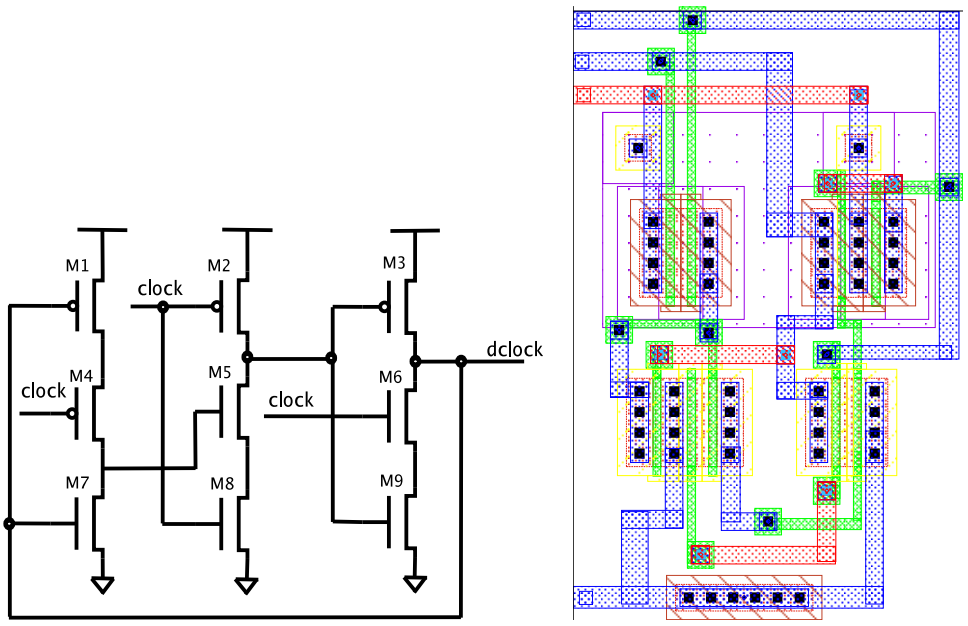
design process. The bee colony optimization algorithm was carefully selected as it has the capability to converge to solutions in reasonable time using minimal computational resources while dealing with complex circuits with large number of parameters.

A. Proposed Design Flow

The design flow proposed in this paper is shown in Fig. 6. As the input specifications are received by the designer the initial logical design is created. The physical layout is created based on this design when it meets specifications. The usual technology rule checks (DRC), layout versus schematic (LVS) and parasitic extraction (RCLK) steps are then performed. As noted previously, the RCLK components have tremendous effect on high frequency AMS circuits. As most of the time the circuit fails specifications, extra steps are introduced to bring the design back to spec without iteratively going back to the physical layout or logical design.

The resulting netlist which includes all parasitics and circuit elements is parameterized. The design parameters are selected among the key components of the PLL circuit. LC-VCO transistors NM_1 and NM_2 (Fig. 4(a)) widths are set to W_{nLC} and PM_1 and PM_2 widths are set to W_{pLC}. The divider (Fig. 5(a)) transistor widths are parameterized separately: W_{n1Div} for M_5, W_{n2Div} for M_6, W_{n3Div} for M_7, W_{n4Div} for M_8, W_{n5Div} for M_9, W_{p1Div} for M_1, W_{p2Div} for M_2, W_{p3Div} for M_3, and W_{p4Div} for M_4. Since the phase detector has too many transistors, this component circuit is parameterized after dividing it into three logical portions. The parameters are distributed between two flip-flops and the AND gate. D-flip-flop DFF_1: W_{npd1} and W_{ppd1}, D-flip-flop DFF_2: W_{npd2} and W_{ppd2}, and AND gate: W_{npd3} and W_{ppd3} (Fig. 2(a)). The charge pump (Fig. 3(a)) is also divided into two different portions. Since the current mirror transistors need to be larger size than the logic transistors of the circuit, the charge pump inverter 1&2 transistors M_1, M_2, M_3, and M_4 are set to W_{nCP1} and W_{pCP1}, and the current mirror transistors M_5, M_6, M_7, and M_8 are set to W_{nCP2} and W_{pCP2}. This results in a total of 21 parameters. The ranges for each parameter are shown in Table III.

The parameters considered are usually up to the designer. In the worst case, they can be the same parameters that will be used for future verification for process variation analysis. The metamodels are created based on the sampling done on the design space. Each metamodel is the direct output for a single specification of the circuit such as power, locking time, phase noise, frequency etc. The optimization is then conducted on the designed metamodel. It should be noted that the optimization time is not directly linked to the simulation time. Hence, the optimization algorithm can be thorough and more computationally expensive since the metamodel itself is computationally cheap.



(a) Schematic of divide by 2 circuit. (b) Physical layout of the divide by 2 circuit.

Fig. 5. Design of divide by 2 circuit.

B. Metamodel Creation

The polynomial metamodel generation steps are shown in Fig. 7. To maximize the accuracy of the model an appropriate sampling technique should be identified. The current paper uses Latin Hypercube Sampling (LHS) as it is accurate and yet fast [8]. The sample data generation stage is the slowest part of the metamodeling process. The accuracy of the metamodel is dependent on the amount of simulations which is limited by the available simulation budget (time-wise). The Latin Hypercube sampling (LHS) is conducted on the number of parameters that exist in the parameterized netlist with parasitics. LHS is a semi-uniform statistical technique that divides the design space space into "Latin" squares which are then sampled randomly. The Latin squares provide more uniform distribution of samples than Monte Carlo sampling which takes random points from the design space with a given distribution for each present parameter.

Latin hypercube sampling was first introduced in [25]. Let $P = P_{jk}$ be an $N \times K$ matrix, where each column of P is an independent random permutation of $1, 2, \cdots, N$. Moreover, let $\xi_{jk}(j = 1, \cdots, N)$ be NK independent and identically distributed over $U[0, 1]$ (uniformly distributed on [0,1]) random variables which are also independent of P. Then the LHS sample matrix X_{jk} is defined by the following:

$$X_{jk} = F_k^{-1}(N^{-1}(p_{jk} - 1 + \xi_{jk})), \tag{1}$$

where F is the joint distribution of the random vector of parameters X. Since the Latin squares are picked randomly from the design space for the number of simulations desired, the total amount of combinations for same LHS design can be quite high: $(M!)^{N-1}$ where M is the number of divisions for each of N parameters to create the Latin square.

Two different LHS sizes are constructed: a large one for a training set and a small set for verification. LHS points (sets of design parameters) are then imported in a script that runs the SPICE simulations. Since the number of simulations is directly linked to the number of different types of analysis that are performed to calculate the specifications, the number of simulations translates into the amount of time it takes to generate the metamodel and is calculated as follows:

$$T_{Meta} = \left(\frac{N_s}{i}\right) \times \sum T_{s_i}. \tag{2}$$

where T_{s_i} is average time to perform a separate simulation for each i analysis and N_s is the total number of simulations.

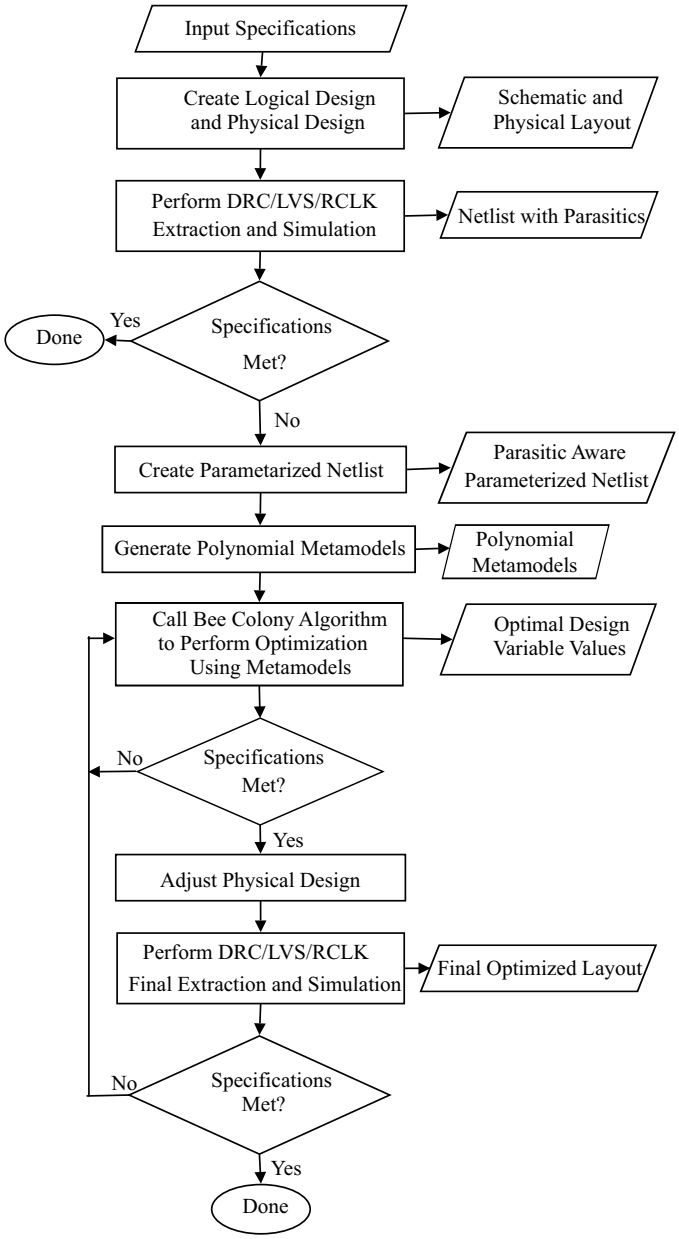


Fig. 6. The proposed ultra-fast design optimization flow that uses polynomial metamodels and bee colony optimization.

Polynomial regression is then performed on the training data set. In this work we consider only polynomial metamodels, hence the accuracy is directly related to the maximum number of coefficients that can be fit in the model given the available data. Since more than one function can be generated to fit the generated data, different order polynomials are considered since regression is not a time consuming process. Partial polynomial functions of order 1 through 6 are considered, since the full polynomial function would result in a very large amount of coefficients for 21 variables.

A stepwise regression method [26] is used to filter out the coefficients that do not contribute to the function's outcome, since it has a chance to exclude some terms which are not statistically needed, resulting in shorter and just as 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. It 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 [26] 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

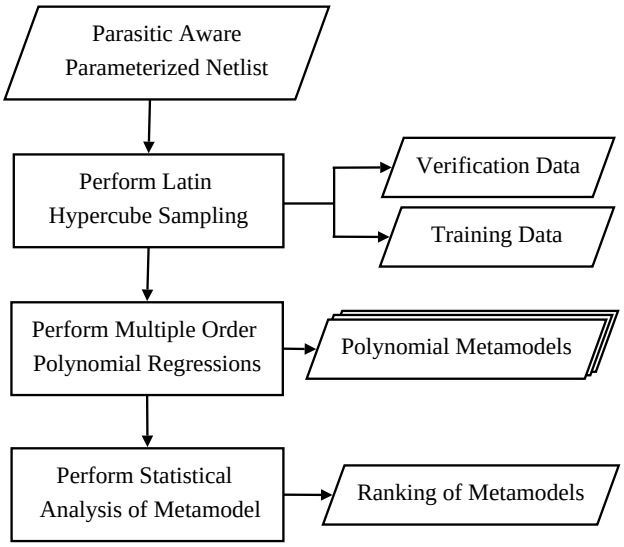


Fig. 7. The proposed approach for metamodel generation.

the model, the null hypothesis is that the term can be ignored, so if there is insufficient evidence to reject the null hypothesis, the term is removed from the model. The method concludes when no more improvements can be made 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 since it changes the order in which terms are moved in and out.

As the manufacturing price of the circuit is very high, it is essential to produce the most accurate metamodel for that design to be manufactured with the lowest tolerance for error. To pick the most accurate function from the pool of functions the RMSE (Root Mean Square Error) and R^2 (coefficient of determination) statistics are generated. Since the statistical data only calculates the goodness of fit to the training data set, the verification LHS input parameters are used and the output of each function is compared to the actual (simulation) data by using the same statistical method. This step ensures that the function is not over-fitted and shows how well the function under consideration performs on the parameter values other than training data. The best metamodeling function is selected from the best RMSE and R^2 values for both data sets.

RMSE and R^2 are the metrics used for goodness of fit in our metamodeling approach. The RMSE is derived from the Sum of Square Errors (SSE):

$$RMSE = \sqrt{\left(\frac{1}{N}\right) SSE}, \tag{3}$$

$$= \sqrt{\left(\frac{1}{N}\right) \sum_{k=1}^{N} (y(x_k) - \hat{y}(x_k))^2}, \tag{4}$$

where N is the number of simulation points, y is the actual simulation result values and $\hat{y}$ are the results of the metamodel at the same location as the simulation point x_k. R^2 predicts the probability that a future result is accurately predicted by the model. R^2 ranges from 0 to 1, where 1 is the best probability value. However, R^2 cannot account for over-fitting of the model. Hence, the adjusted R^2 which accounts for the number of explanatory terms in a function, R^2_{adj}, is used [26]. Both R^2 and R^2_{adj} for different orders of the polynomial metamodels for settling time are shown in Fig. 8. The number of coefficients that are generated for each order of the polynomial metamodel is shown in Fig. 9. In can be seen that the R^2 value and R^2_{adj} are nearly equal to 1 when the order reaches 5. The number of coefficients that represent the metamodel at those orders is equal to the number of simulation data points (100). This means that the model is over fitted, therefore for the metamodel that represents settling time, a polynomial order of 4 will be used. R^2 is calculated using the following expression:

$$R^2 = \frac{SSE}{SST} = \frac{\sum_k (y(x_k) - \hat{y}(x_k))^2}{\sum_k (y(x_k) - \bar{y})^2}, \tag{5}$$

where $\bar{y}$) is the mean value of the simulation responses at the N sample points. R^2_{adj} is calculated by including the effect of the degrees of freedom in the data:

$$R^2_{adj} = 1 - \left(\frac{N-1}{N-p}\right)\left(1 - R^2\right), \tag{6}$$

where p is the total number of terms in the polynomial metamodel (including the constant term).

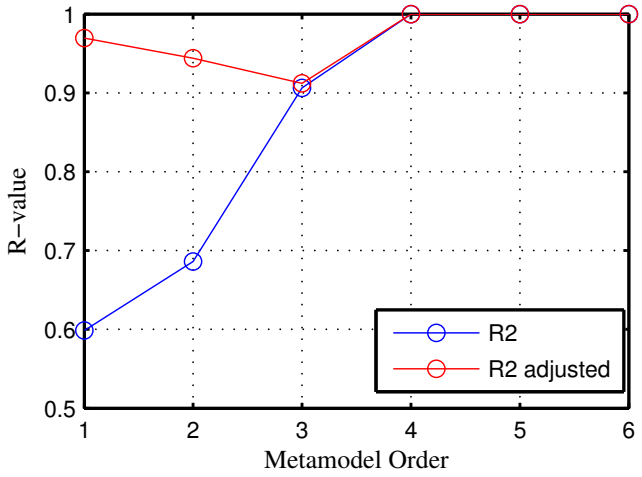


Fig. 8. Generated R^2 and R^2_{adj} for various orders of the polynomial metamodels for frequency.

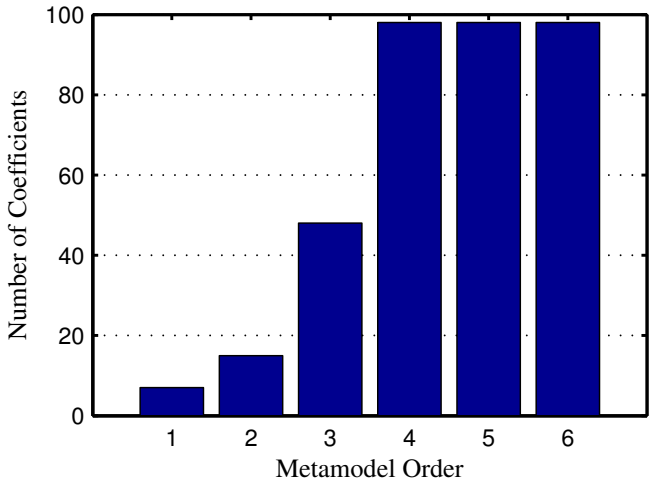


Fig. 9. The number of coefficients corresponding to the order of the generated metamodels for frequency.

The statistics for all polynomial metamodels of the selected FoMs for the PLL that were selected for optimization are presented in Table I. It is interesting to note that metamodels of higher order than 2 primarily show better results. This is an important observation because the commonly used Response Surface Methods (RSM) for optimization only used a local representation of the data that is at most quadratic. In contrast, the metamodel representation is not only global but of high degree as well.

C. Bee Colony Optimization Algorithm

In this paper, the bee colony optimization (BCO) algorithm is investigated for nanoscale PLL design as the parasitic-aware netlist simulation time is excessive for efficient design space exploration. The BCO algorithm is based on the natural behavior of honey bees for finding the best food source. The artificial bee colony divides bees

TABLE I
SELECTED METAMODEL STATISTICS FOR EACH FOM OF PLL CIRCUITS

Metamodels	Order	R^2	R^2 adjusted	Coefficients in model	RMSE fit (W)
Frequency	3	0.9064	0.9119	48	33.397 MHz
Power	3	0.9985	0.9980	56	0.157 mW
Settling Time	4	0.8558	0.8748	57	9.96 ns
Horizontal Jitter	3	0.7041	0.8452	34	36.37 ps
Vertical Jitter	3	0.7125	0.7968	41	1.6 mV

into three categories: onlookers, scouts, and workers. The algorithm starts with bees being divided equally between onlookers and worker bees only. The state diagram in Fig. 10 shows the states and transitions between stages that the bees can go though. An initial random solution is assigned to worker bees. Worker bees then search for food at the known random location. When bees return to the hive, the information is shared among the bees by performing a wiggly dance on the hive 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. If the worker bee does not get enough food source it becomes unemployed and waits for a dance that it will like with probability P_{on}. Then it goes to the location of proximity to the advertised location. If the food source is not as good as expected it goes on searching in other random locations by becoming a scout bee. If the scout finds a better food source, it returns to the hive and becomes a worker bee again by wiggle dancing and trying to recruit onlookers.

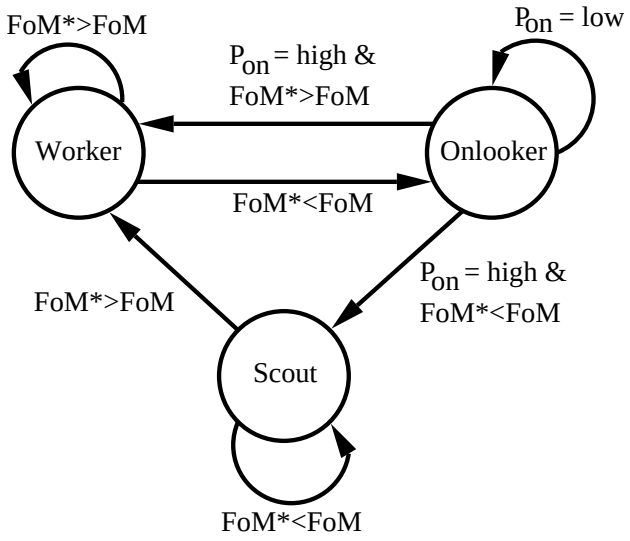


Fig. 10. Bee transition states in the beehive implemented in the bee colony algorithm.

The internals of the meta-heuristic BCO algorithm which is used for fast design space exploration of the PLL in this paper are described in Algorithm 1. The proposed algorithm is a maximization approach. Due to the random behavior of bees it can leave local maxima and potentially find the global maximum, provided a sufficiently large number of iterations is performed. The FoMs used in this algorithm are presented in Section VI.

VI. EXPERIMENTAL RESULTS

To accommodate the metamodeling approach, multiple CAD tools are used. The data flows from/to different tools at different steps of the process. The initial (logical) design is done using SPICE. LHS generation is conducted in MATLAB which in turn creates the LHS design matrix for all parameters and generates the script file to run the SPICE simulations. As the simulations are run, the data is fed back to MATLAB where the metamodel generation,

Algorithm 1 Proposed Bee Colony Optimization Algorithm.

Initialize maximum iterations $\leftarrow max_i$.
Set the boundaries for each parameter $P(i) \leftarrow [min, max]$.
$NumberBees \leftarrow$ **Define** the number of bees.
$buffer \leftarrow$ **Define** number of close worker bees dispersal.
Initialize a matrix as follows: $bee_{matrix}(3, NumberBees) \leftarrow [workers, onlookers, scouts]$.
Set bee_{matrix} first half to be workers and other onlookers.
Initialize food sources.
while $(Counter \leq max_i)$ **do**
  for each i from 1 to $NumberBees$ **do**
    if $(bee_{matrix}(1, i) == 1)$ **then**
      (1) **Send** worker bee to a random known food source.
      Calculate Power(i), Jitter$_{h/v}$(i) using metamodels.
      Calculate the proposed FoM of the PLL.
      if (current FoM is better than the previous FoM) **then**
        Update result and location.
      else
        Convert bee to onlooker.
      end if
    else if $(bee_{matrix}(1, i) == 1)$ **then**
      (2) **Send** onlooker bee.
      Calculate $P_{on} \leftarrow$ probability of the food source being good
      if (P_{on} is high) **then**
        $P \leftarrow (P_{min} + random(1) \times P_{max}) \times buffer$.
        Send onlooker to random location for each design parameter P.
        Calculate the FoM.
        if (current FoM is better than the previous FoM) **then**
          Update result and location.
          Convert bee to worker.
        else
          Convert bee to scout.
        end if
      end if
    else
      (3) **Send** scout bee.
      Pick the best result as $best_r$.
      $P \leftarrow P_{min} + random(1) \times P_{max}$.
      Send the scout to random location for each P.
      if (current FoM is better than the previous FoM) **then**
        Update the result.
        Convert bee to worker.
      end if
    end if
    if (current FoM is better than previous FoM) **then**
      Update result and location.
    end if
  end for
  $Counter \leftarrow Counter + 1$.
end while
Return result and location.

statistical analysis and optimization are conducted. The optimized values are transferred to SPICE where the final adjustments are done on the physical layout and verification simulations are run.

The proposed optimization is performed under a constraint of locking time. The locking time is a metric that shows that the PLL circuit really works as expected and is able to lock within reasonable time. The optimization target is that the frequency is within 0.5% of the specification. The aim is for a 2.6 GHz output frequency of the PLL. The FoM that needs to be maximized is calculated for every arising combination of parameters varied for optimization. In this work the following FoM for the PLL is introduced, to ensure that mutual conflicting objectives are met during the optimization:

$$FoM_{\text{PLL}} = \left(\frac{1}{\text{Power} \times \text{Jitter}_h \times \text{Jitter}_v} \right), \tag{7}$$

where P, J_h, and J_v are the power, horizontal jitter, and vertical jitter, respectively. The maximization of this FoM will lead to a PLL design that will have minimized power and jitter while able to achieve a lock.

The BCO progression over the number of iterations is shown in Fig. 11. Each run of the optimization is slightly different from the others due to the random effect of the bee transitions and food search in the given design space.

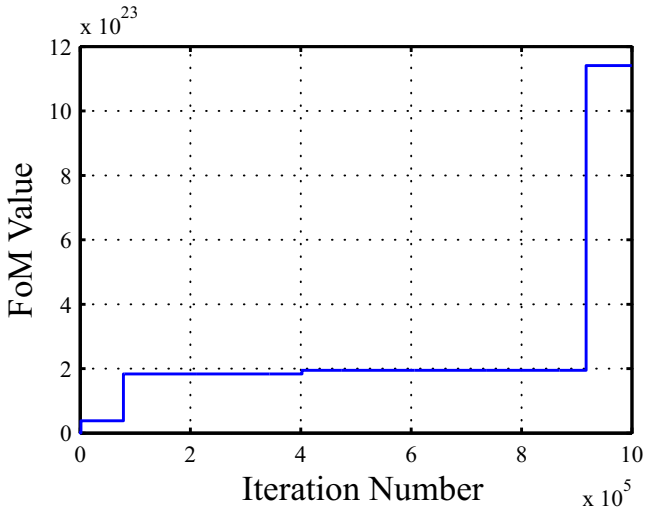


Fig. 11. Results of the BCO algorithm progression for the selected FoM for up to 10^6 iterations.

The results of the BCO are shown in Table II. The final optimization shows more than 90% power improvement and very high vertical jitter improvement over the baseline design with no area penalty which is due to the layout configuration and the selected parameters that do not affect the final layout considerably. The final optimized responses for each parameter of the PLL are shown in Table III.

TABLE II

POWER AND JITTER OF THE PLL BEFORE AND AFTER OPTIMIZATION.

Metric	Before Optimization	After Optimization	Improvement
Power	9.29 mW	0.87 mW	90.6%
Jitter Vertical	168.35μV	3.28 nV	$\sim$100%
Jitter Horizontal	189 ps	180 ps	4.8%
Area	525×326 μm^2	525×326 μm^2	0%

Careful consideration was given to the initial physical design to provide enough space for transistor resizing. Even though it might affect the compactness of transistors in the initial layout in comparison to the optimal placed physical design, the speed up that this design flow provides outweighs the area savings. The final physical design of the PLL that uses the optimized parameters is shown in Fig. 12. Most of the parameters used are located in the lower left corner of the layout, which includes the phase detector, charge pump, loop filter, and divider. The

TABLE III

PLL CIRCUIT PARAMETERS WITH EACH PARAMETER RANGE. THE FINAL COLUMN ALSO SHOWS THE OPTIMIZED VALUES FOR EACH PARAMETER.

Circuit	Parameter	Min (m)	Max (m)	Optimal Value (m)
Phase Detector	W_{ppd1}	400n	2μ	1.66μ
	W_{npd1}	400n	2μ	1.11μ
	W_{ppd2}	400n	2μ	784n
	W_{npd2}	400n	2μ	689n
	W_{ppd3}	400n	2μ	1.54μ
	W_{npd3}	400n	2μ	737n
Charge Pump	W_{nCP1}	400n	2μ	1.24μ
	W_{pCP1}	400n	2μ	1.35μ
	W_{nCP2}	1μ	4μ	1.35μ
	W_{pCP2}	1μ	4μ	2.88μ
LC-VCO	W_{nLC}	3μ	20μ	18.62μ
	W_{pLC}	6μ	40μ	37.48μ
Divider	W_{p1Div}	400n	2μ	1.65μ
	W_{p2Div}	400n	2μ	1.54μ
	W_{p3Div}	400n	2μ	1.38μ
	W_{p4Div}	400n	2μ	1.96μ
	W_{n1Div}	400n	2μ	1.09μ
	W_{n2Div}	400n	2μ	1.17μ
	W_{n3Div}	400n	2μ	1.29μ
	W_{n4Div}	400n	2μ	1.95μ
	W_{n5Div}	400n	2μ	536n

LC-VCO takes up most of the area of the PLL, but since the inductor and capacitors have not been parameterized, there was no area penalty to this design optimization from the initial physical design.

VII. CONCLUSION AND FUTURE RESEARCH

This paper has provided a detailed design flow that uses metamodels to speed up the design process for AMS components. A 2.6 GHz PLL circuit that was designed using 180 nm nano-CMOS technology was used as case study. The physical layout was parameterized using 21 design parameters and optimized using the bee colony optimization algorithm. It was demonstrated that the algorithm is suitable for AMS circuit optimization by use of metamodels and has provided significant optimization improvements in FoM characteristics. The optimization was conducted on 5 metamodels which were created on only 100 sampling points and have been shown to provide excellent characterization of the PLL circuit with high accuracy. The final outcome of the design flow was 90% power savings and an average of 52% jitter minimization. The final physical layout of the process is shown to have 0% area penalty from the initial design. Compared to an exhaustive search of the 21-dimensional design space, with 10 samples per parameter, 10^{21} simulations would be required. The time savings are enormous ($\approx 10^{20}\times$ simulation time). In future research we will investigate metamodels other than polynomial. Since the accuracy of the metamodel is essential, it would be interesting to see the behavior of different kinds of equations and algorithms and their accuracy applied to complex circuits with very large parameter sets (100+).

ACKNOWLEDGMENTS

This research was partially supported by SRC award P10883 and NSF awards CNS-0854182 and CCLI-0942629. The preliminary results of this archival journal paper is presented in the following blind-review conference: [27].

REFERENCES

[1] D. Ghai, S. P. Mohanty, and E. Kougianos, "Design of Parasitic and Process-Variation Aware Nano-CMOS RF Circuits: A VCO Case Study," *IEEE Trans. VLSI Syst.*, vol. 17, no. 9, pp. 1339–1342, 2009.

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

[3] S. Basu, B. Kommineni, and R. Vemuri, "Variation-Aware macromodeling and Synthesis of Analog Circuits using Spline Center and Range Method and Dynamically Reduced Design Space," in *Proc. Int. Conf. VLSI Des., VLSID*, 2009, pp. 433–438.

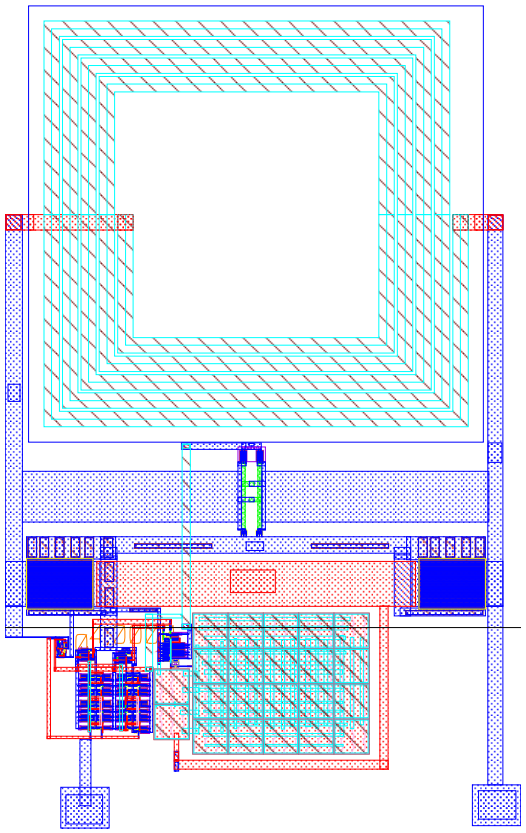


Fig. 12. Final layout of the optimized PLL for 180nm nano-CMOS technology.

[4] J. Wang, X. Li, and L. T. Pileggi, "Parameterized Macromodeling for Analog System-Level Design Exploration," in *Proceedings of the 44th Design Automation Conference*, 2007, pp. 940–943.

[5] M. Ding and R. Vemuri, "Efficient Analog Performance Macromodeling via Sequential Design Space Decomposition," in *Proc. Int. Conf. VLSI Des., VLSID*, 2006, pp. 553–556.

[6] A. Agarwal, G. Wolfe, and R. Vemuri, "Accuracy Driven Performance macromodeling of Feasible Regions during Synthesis of Analog Circuits," in *Proc. Great Lakes Symp. VLSI, GLSVLSI*, 2005, pp. 482–487.

[7] O. Garitselov, S. P. Mohanty, and E. Kougianos, "Fast optimization of nano-CMOS mixed-signal circuits through accurate metamodeling," in *Proceedings of the 12th International Symposium on Quality Electronic Design*, 2011, pp. 405–410.

[8] ——, "Nano-CMOS Mixed-Signal Circuit Metamodeling Techniques: A Comparative Study," in *Proc. Int. Symp. Elec. Des.*, 2010, pp. 191–196.

[9] D. A. Mathaikutty and S. K. Shukla, *Metamodeling Driven IP Reuse for System-on-chip Integration and Microprocessor Design*. Artech House, 2007.

[10] K.-T. Fan, R. Li, and A. Sudjianto, *Design and Modeling for Computer Experiments*. 23-25 Blades Court, London SW15 2NU, UK: Chapman and Hall/CRC, 2006.

[11] R. Samanta, J. Hu, and P. Li, "Discrete Buffer and Wire Sizing for Link-Based Non-Tree Clock Networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 7, pp. 1025–1035, July 2010.

[12] A. Lamecki, L. Balewski, and M. Mrozowski, "Towards Automated Full-Wave Design of Microwave Circuits," in *Proceedings of the 17th International Conference on Microwaves, Radar and Wireless Communications*, 19-21 2008, pp. 1–2.

[13] W. Dong, Z. Feng, and P. Li, "Efficient VCO phase macromodel generation considering statistical parametric variations," in *Proceedings of the IEEE/ACM international conference on Computer-aided design*, 2007, pp. 874–878.

[14] J. L. Wong, A. Davoodi, A. Khanderwal, A. Srivastava, and M. Potkonjak, "A Statistical Methodology for Wire-Length Prediction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 7, pp. 1327–1336, July 2006.

[15] S. Roy and C. C.-P. Chen, "SmartSmooth: A Linear Time Convexity Preserving Smoothing Algorithm for Numerically Convex Data with Application to VLSI Design," in *Proceedings of the Asia and South Pacific Design Automation Conference*, 2007, pp. 559–564.

[16] A. Sathanur, R. Chakraborty, and V. Jandhyala, "Accurate Statistical Analysis of a Differential Low Noise Amplifier Using a Combined SPICE-Field Solver Approach," in *IEEE International Symposium on Circuits and Systems, 2008. ISCAS 2008.*, May 2008, pp. 884–887.

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

[18] R. A. Thakker, M. S. Baghini, and M. B. Patil, "Low-Power Low-Voltage Analog Circuit Design Using Hierarchical Particle Swarm Optimization," in *Proceedings of the 22nd International Conference on VLSI Design*, 2009, pp. 427–432.

[19] Y. Delican, R. A. Vural, and T. Yildirim, "Artificial Bee Colony Optimization based CMOS Inverter Design Considering Propagation

Delays," in *Proceedings of the XIth International Workshop on Symbolic and Numerical Methods, Modeling and Applications to Circuit Design (SM2ACD)*, 2010, pp. 1–5.
[20] S. Sabat, K. Kumar, and S. Udgata, "Differential evolution and swarm intelligence techniques for analog circuit synthesis," in *Proceedings of the World Congress on Nature Biologically Inspired Computing*, 2009, pp. 469–474.
[21] N. Karaboga, S. Kockanat, and H. Dogan, "Parameter determination of the schottky barrier diode using by artificial bee colony algorithm," in *Proceedings of the International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, 2011, pp. 6–10.
[22] S. Gupta, "Phase-Locked Loops," *Proceedings of the IEEE*, vol. 63, no. 2, pp. 291–306, February 1975.
[23] Z. Wang, "An Analysis of Charge-Pump Phase-Locked Loops," *IEEE Transactions on Circuit and Systems-I: Fundamental Theory and Applications*, vol. 52, no. 10, pp. 2128–2138, October 2005.
[24] F. Gardner, "Charge-Pump Phase-Lock Loops," *IEEE Transactions on Communications*, vol. 28, no. 11, pp. 1849–1858, November 1980.
[25] M. D. McKay, R. J. Beckman, and W. J. Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, vol. 21, no. 2, pp. 239–245, May 1979.
[26] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*. John Wiley and Sons, Inc, 2006.
[27] O. Garitselov, S. P. Mohanty, E. Kougianos, and P. Patra, "Bee Colony Inspired Metamodeling Based Fast Optimization of a Nano-CMOS PLL," in *Proceedings of the 2nd IEEE International Symposium on Electronic System Design (ISED)*, 2011.

AUTHORS' BIOGRAPHIES

Oleg Garitselov

Oleg Garitselov received Bachelors and Masters degree from Baumann Moscow State University. In February 2012, he defended his dissertation for a Ph.D. (Computer Science and Engineering) at the University of North Texas (UNT). He is currently a research assistant for Semiconductor Research Corportation (SRC) funded project. His current research includes approaches for low-power design and test. He is an author of 6 peer-reviewed publications in this area of research. His research is in nanoscale mixed-signal circuit design and optimization. He is an active reviewer of many international conferences. He has been named the Outstanding PhD Student in Computer Science and Engineering for the year 2011-2012.

Saraju P. Mohanty

Saraju P. Mohanty is currently an Associate Professor at the Department of Computer Science and Engineering, University of North Texas, and the Director of the NanoSystem Design Laboratory (NSDL, `http://nsdl.cse.unt.edu`). He obtained Ph.D. in Computer Science and Engineering from the University of South Florida in 2003, Masters degree in Systems Science and Automation from the Indian Institute of Science, Bangalore, India in 1999, and Bachelors degree in Electrical Engineering from Orissa University of Agriculture and Technology, Bhubaneswar, India in 1995. His research is in "Design and CAD for Low-Power High-Performance Nanoscale Digital and Analog/Mixed-Signal VLSI". Prof. Mohanty's research is funded by National Science Foundation and Semiconductor Research Corporation. He is an author of 130+ peer-reviewed top-notch journal and conference publications. The publications are well-received by world-wide peers with a total of 1100+ citations resulting in an H-index of 19 and i10-index of 36 (from Google Scholar). He serves on the organizing/program committee of several international conferences and editorial board of several international journals. Prof. Mohanty is a senior member of IEEE and ACM.

Elias Kougianos

Elias Kougianos is currently an Associate Professor in the Department of Electrical Engineering Technology, at the University of North Texas (UNT), Denton, TX. From 1988 through 1997 he was with Texas Instruments, Inc., in Houston and Dallas, TX. Initially he concentrated on process integration of flash memories and later as a researcher in the areas of Technology CAD and VLSI CAD development. In 1997 he joined Avant! Corp. (now Synopsys) in Phoenix, AZ as a Senior Applications engineer and in 2001 he joined Cadence Design Systems, Inc., in Dallas, TX as a Senior Architect in Analog/Mixed-Signal Custom IC design. He has been at UNT since 2004. His research interests are in the area of Analog/Mixed-Signal/RF IC design and simulation and in the development of VLSI architectures for multimedia applications. He is author or co-author of over 60 peer-reviewed journal and conference publications. He is a senior member of IEEE.