Bee Colony Inspired Metamodeling Based Fast Optimization of a Nano-CMOS PLL

Presenter and Contact Author:

Saraju P. Mohanty

NanoSystem Design Laboratory (NSDL, http://nsdl.cse.unt.edu)
Dept. of Computer Science and Engineering

University of North Texas, Denton, TX 76203, USA.

E-mail: saraju.mohanty@unt.edu

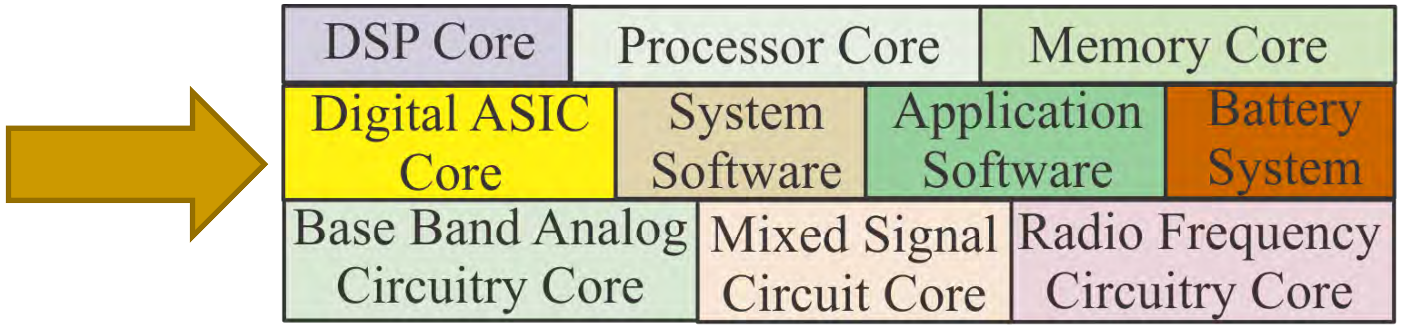
Acknowledgment: This research is supported in part by SRC award P10883 and NSF awards CNS-0854182 and DUE-0942629.

Outline of the Talk

- Introduction and Motivation
- Related Prior Research
- Key Contributions of this Paper
- Proposed Metamodeling Based Design Flow
- Case Study Circuit Design
- Polynomial Metamodel Generation
- Bee Colony Optimization Algorithm
- Experimental Results
- Conclusions

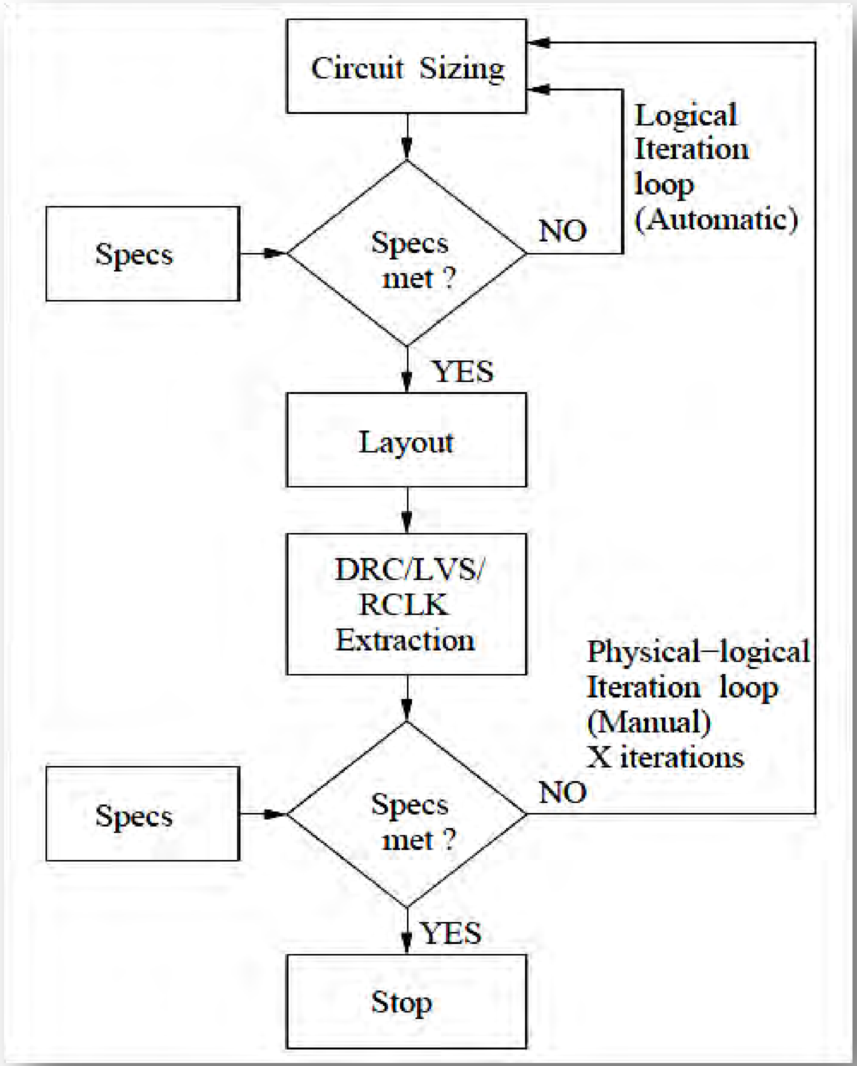
Introduction and Motivation

Analog/Mixed-Signal Systems



- A typical consumer electronics is an Analog/Mixed-Signal System-on-a-Chip (AMS-SoC).
- Individual subsystems can also be mixed-signal, e.g. Phase-Locked Loop (PLL).

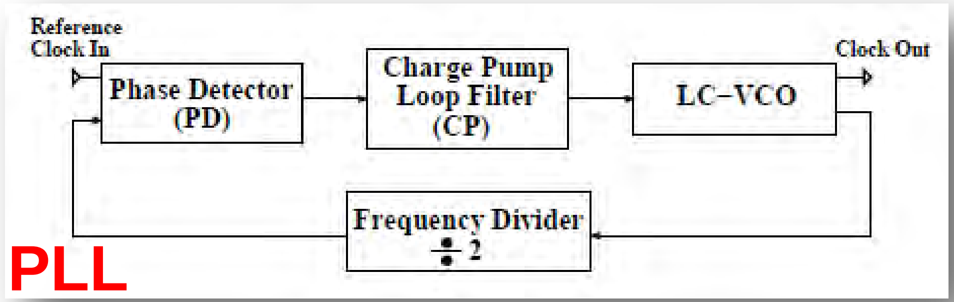
Standard Design Flow



- Standard design flow requires multiple manual iterations on the back-end layout to achieve parasitic closure between front-end circuit and back-end layout.
- Longer design cycle time.
- Error prone design.
- Higher non-recurrent cost.
- Difficult to handle nanoscale challenges.

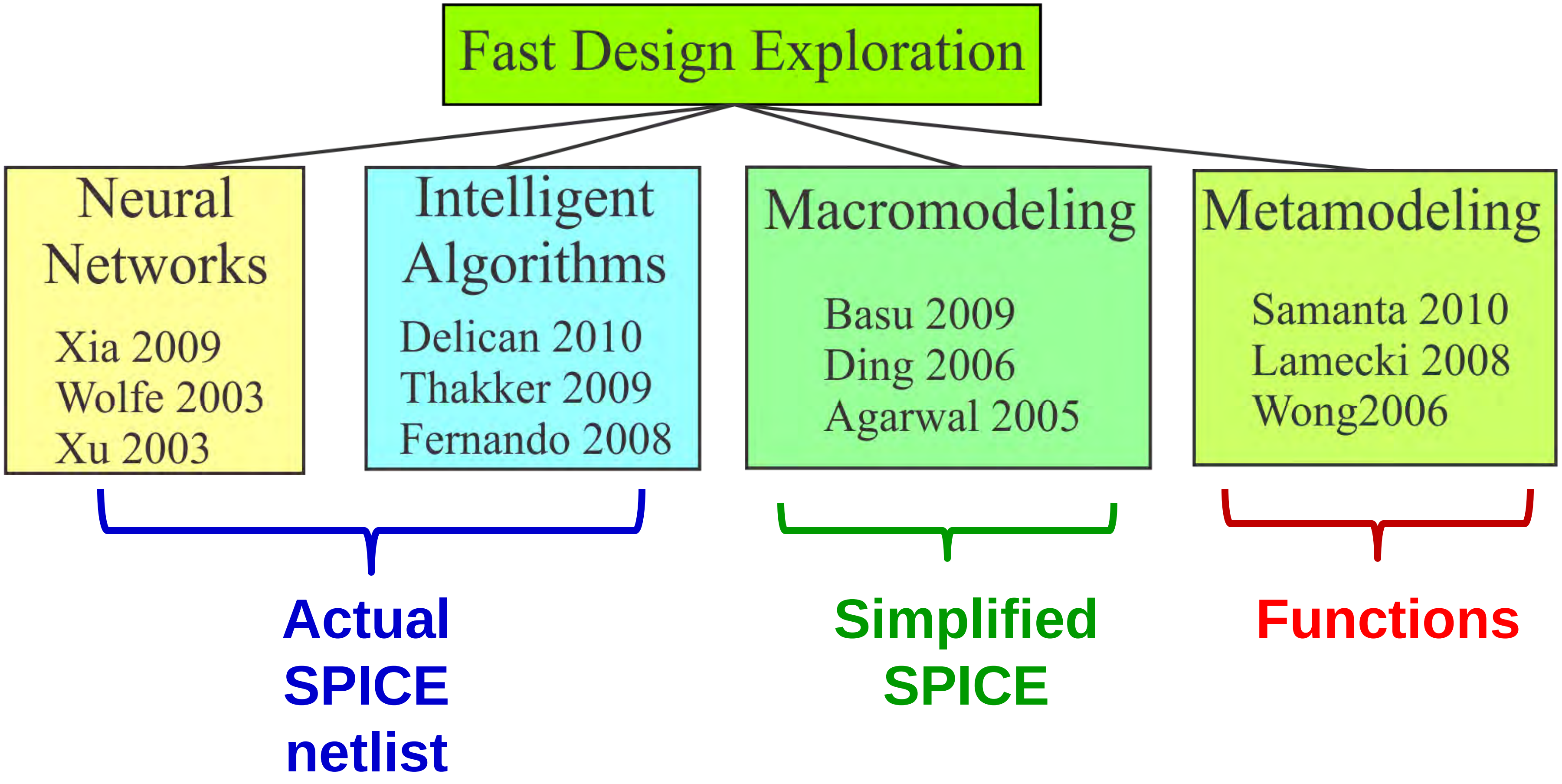
One of the Key Issues

- The simulation time for a Phase-Locked-Loop (PLL) lock on a full-blown parasitic netlist is of the order of many days to weeks!



- **Issues for AMS-SoC components:**
  - How fast can design space exploration be performed?
  - How fast can layout generation and optimization be performed?

Related Prior Research

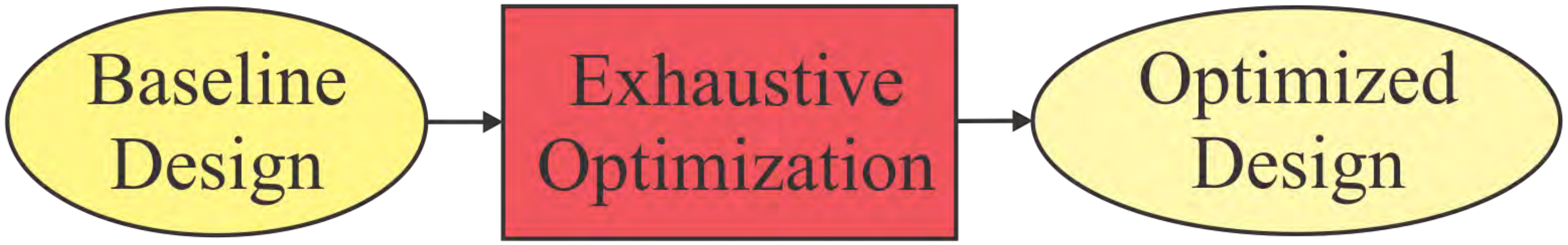


Key Contributions

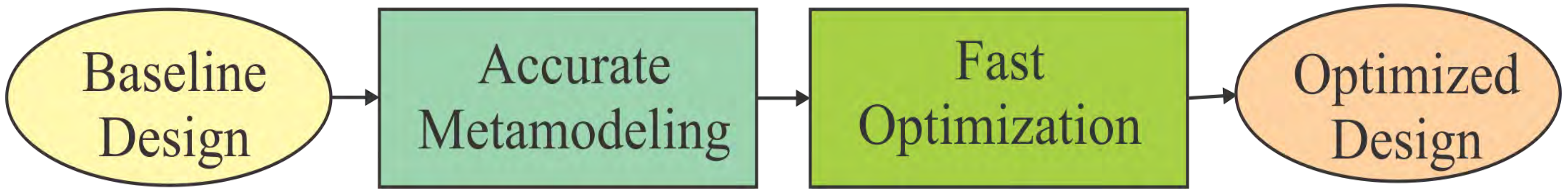
Contributions of This Paper

- Approaches to create accurate, parasitic-aware metamodels for complex nanoscale AMS circuits such as a PLL.
- Accurate models could be created for 21 design parameters from 100 SPICE-level simulations.
- The Bee Colony Optimization (BCO) algorithm is investigated for AMS circuit optimization.
- Demonstrated that the BCO assisted, metamodeling based design flow is orders of magnitude faster than circuit-based approaches.

Fast Design Exploration Through Metamodeling

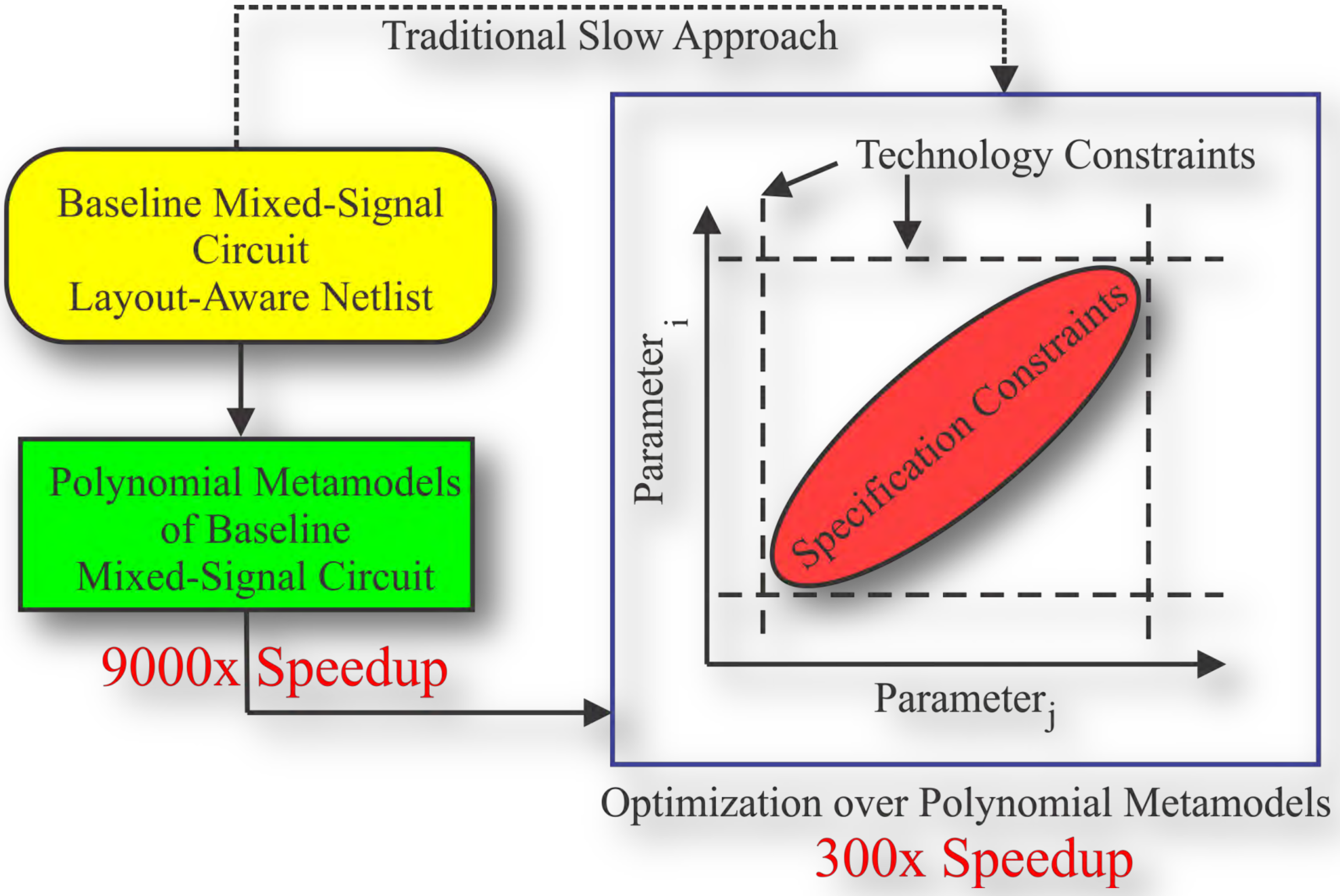


The Traditional Slow Approach



The Metamodel-Based Fast Approach

Two Tier Speed Up


Traditional Slow Approach
Baseline Mixed-Signal Circuit Layout-Aware Netlist
Polynomial Metamodels of Baseline Mixed-Signal Circuit
9000x Speedup
Technology Constraints
Specification Constraints
Parameter_i
Parameter_j
Optimization over Polynomial Metamodels
300x Speedup

Metamodeling vs. Macromodeling

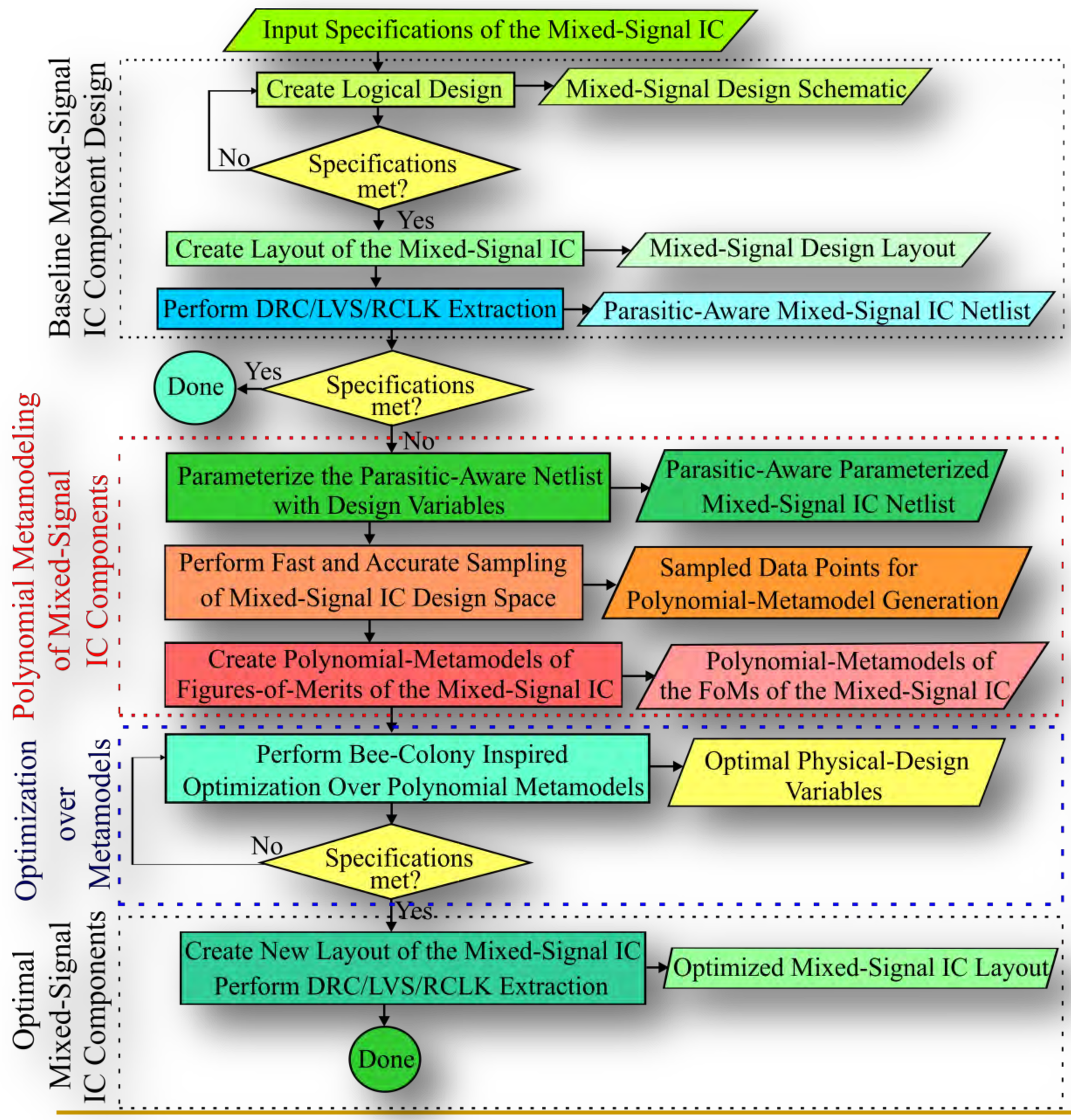
- Macromodeling
  - Simplified version of the circuit.
  - Used in the same simulation tool.
  - Hard to create.
- Metamodeling
  - Mathematical representation of output.
  - Based on prediction equation or algorithm.
  - Language and tool independent.
  - Can be used in different tools such as MATLAB.

Proposed Design Flow

Key Perspective

- Novel design and optimization methodology that will produce robust AMS-SoC components in **one design iteration only** and with minimal (at most two) manual layout steps to improve circuit yield, design cycle time, and reduce chip cost.

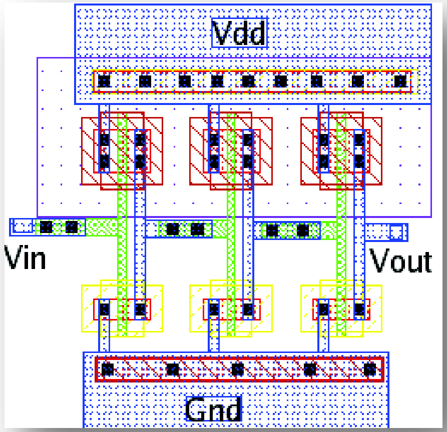
The Proposed Design Flow



Input Specifications of the Mixed-Signal IC
Baseline Mixed-Signal IC Component Design
Create Logical Design
Mixed-Signal Design Schematic
No
Specifications met?
Yes
Create Layout of the Mixed-Signal IC
Mixed-Signal Design Layout
Perform DRC/LVS/RCLK Extraction
Parasitic-Aware Mixed-Signal IC Netlist
Done
Yes
Specifications met?
No
Polynomial Metamodeling of Mixed-Signal IC Components
Parameterize the Parasitic-Aware Netlist with Design Variables
Parasitic-Aware Parameterized Mixed-Signal IC Netlist
Perform Fast and Accurate Sampling of Mixed-Signal IC Design Space
Sampled Data Points for Polynomial-Metamodel Generation
Create Polynomial-Metamodels of Figures-of-Merits of the Mixed-Signal IC
Polynomial-Metamodels of the FoMs of the Mixed-Signal IC
Optimization over Metamodels
Perform Bee-Colony Inspired Optimization Over Polynomial Metamodels
Optimal Physical-Design Variables
No
Specifications met?
Yes
Optimal Mixed-Signal IC Components
Create New Layout of the Mixed-Signal IC Perform DRC/LVS/RCLK Extraction
Optimized Mixed-Signal IC Layout
Done

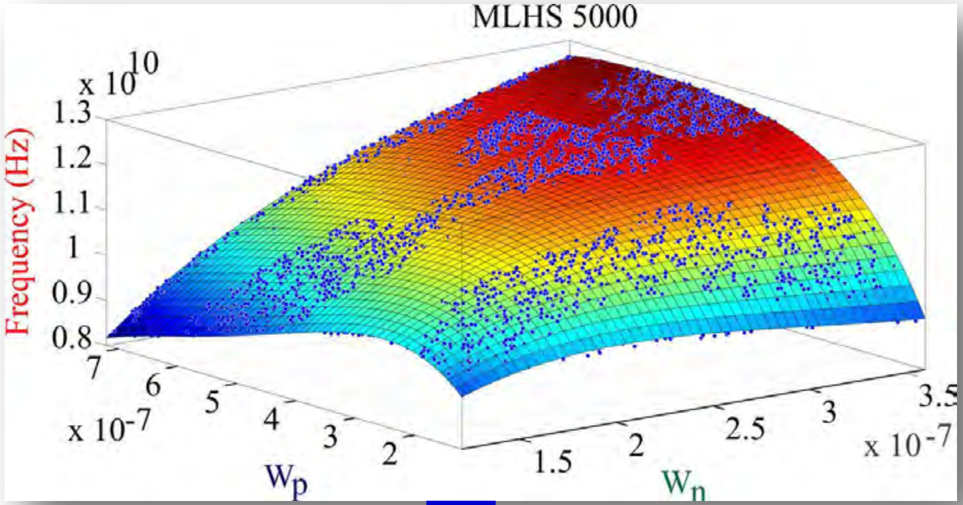
- ❖ Polynomial metamodeling is introduced in the flow.
- ❖ Bee-colony optimization algorithm is introduced.
- ❖ Physical design is performed only 2 times in the proposed design flow.

Polynomial Metamodeling



Actual Circuit (SPICE netlist) of AMS-SoC Components

Statistical Sampling



Polynomial Function Fitting

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

Bee-Colony Optimization: Overview

1. **Initial** food sources are produced for all worker bees.
2. **Do**
   1) Each worker bee goes to a food source and evaluates its nectar amount.
   2) Each onlooker bee watches the dance of worker bees and chooses one of their sources depending on the dances and evaluates its nectar amount.
   3) Determine abandoned food sources and replace with the new food sources discovered by scouts.
   4) Best food source determined so far is recorded.
3. **While** (requirements are met)

Position of a food source → a solution; Nectar amount → Quality of a solution; Number of worker bees → number of solutions in the population.

Bee-Colony Optimization Algorithm

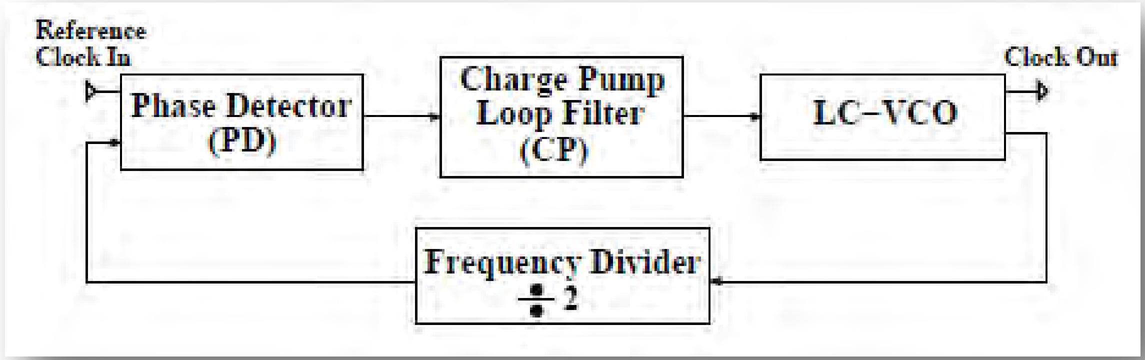
```
1.  Set boundaries for each parameter P(i)[min,max] (i.e. $W_n$, $W_p$, L …).
2.  Initialize NumberOfBees (i.e. # of parameters), beematrix[workers,
    onlookers, scouts] (i.e. 0/1 entries), food sources (i.e. sample point).
3.  while (Counter  $max_i$) do
4.    for each i from 1 to NumberOfBees do
5.      if (beematrix(1, i) == 1) then Send worker bee to a random
        known food source and FoMs using metamodels.
6.        if (better current FoM) then Update result (i.e. FoM) and
          location (i.e. parameter $W_n$, $W_p$, L …).
7.      Else if (beematrix(1, i) == 1) then Send onlooker bee.
8.        Calculate probability that the food source is good.
9.        if (probability is high) then Send onlooker to random location
          for each design parameter P and Calculate the FoM.
10.         if (better current FoM) then Update result and location.
11.       Else Send scout bee and Pick the best result.
12.         Send the scout to random location for each P.
13.       if (current FoM is better than the previous FoM) then
14.         Update the result and Convert bee to worker.
15.     if (better current FoM) then Update result and location.
16. Return result and location.
```

NanoSystem Design Laboratory

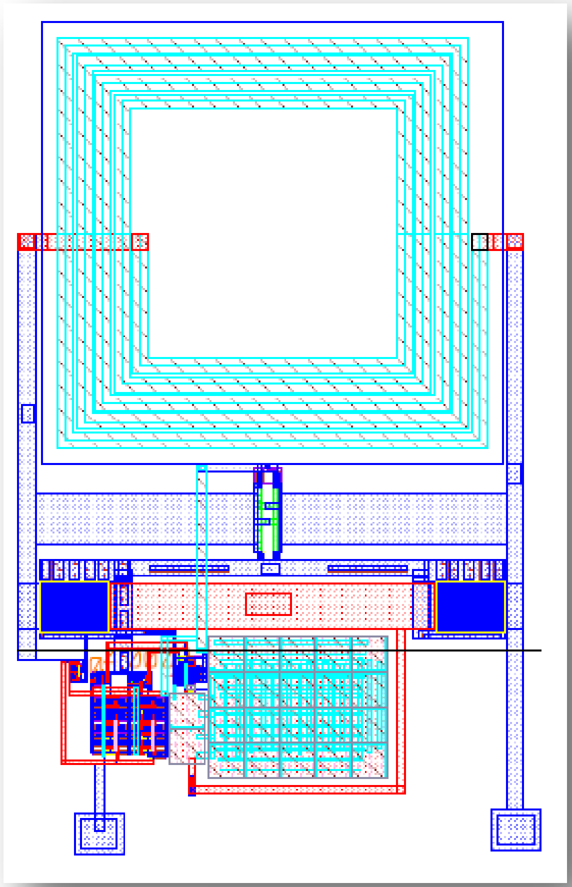
UNT UNIVERSITY OF NORTH TEXAS Discover the power of ideas

Experiments

Case Study Circuit: 180nm PLL



Block diagram of a PLL.

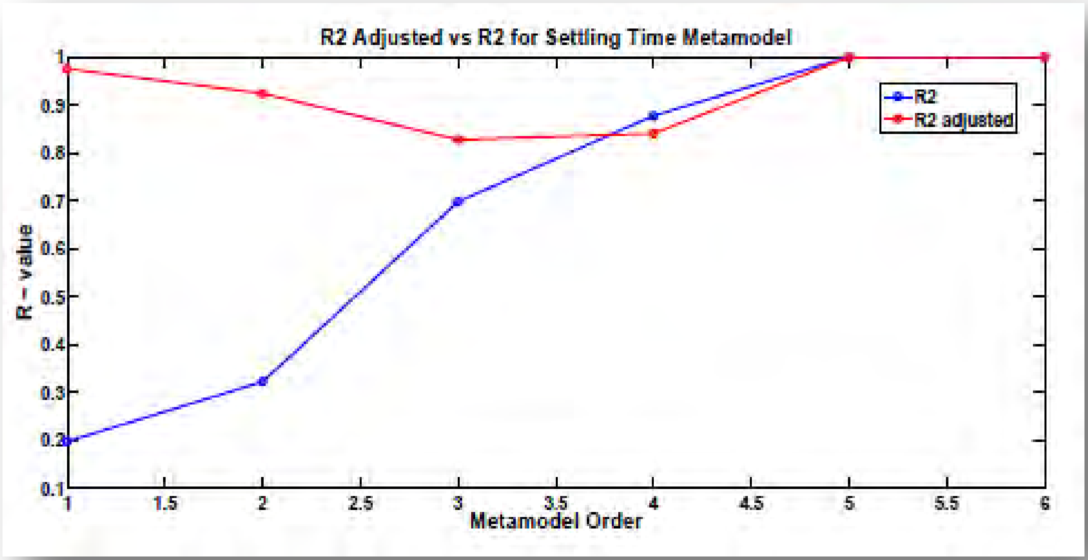


PLL for 180nm.

- PLL circuit is characterized for frequency, power, vertical and horizontal jitter (for simple phase noise), and locking time.
- Metamodels are created for each FoM from same sample set.

Metamodels of the PLL …

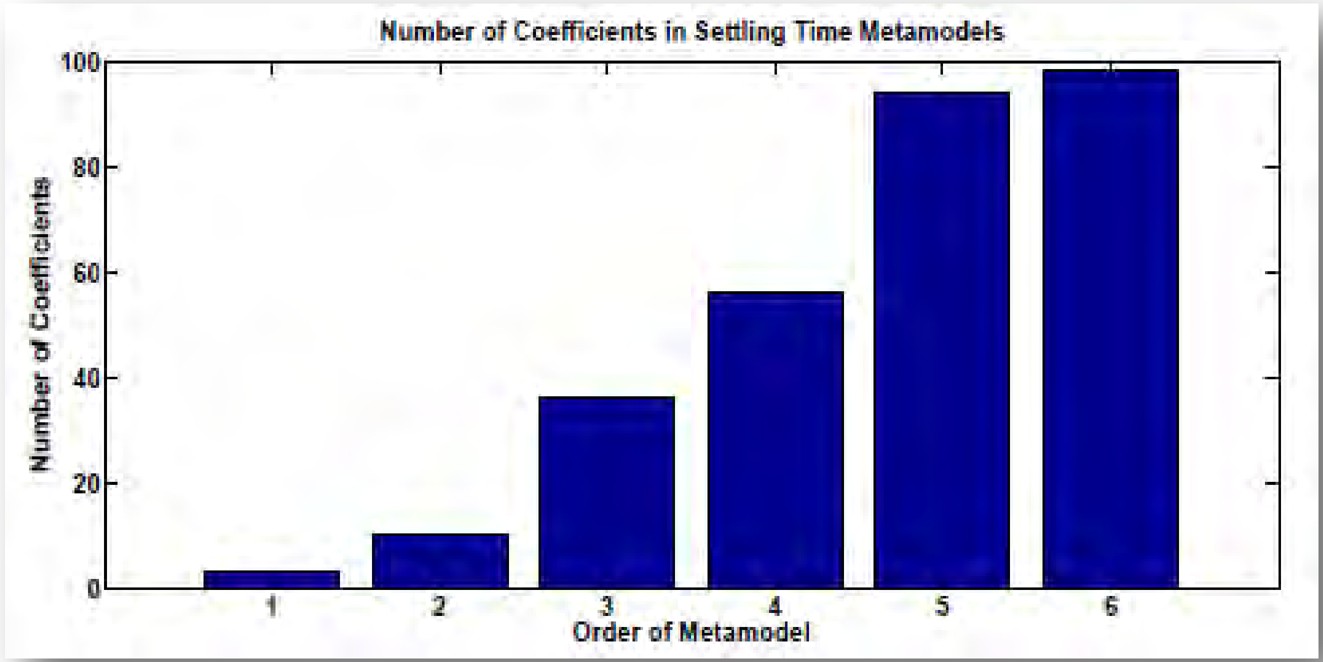
- The PLL circuit is characterized for output frequency, power, vertical and horizontal jitter (to simplify the phase noise calculations), and locking time (or settling time).
- A separate metamodel is created for each FoM from the same sample set.
- The Root Mean Square Error (RMSE) and coefficient of determination R^2 are the metrics used for goodness of fit.



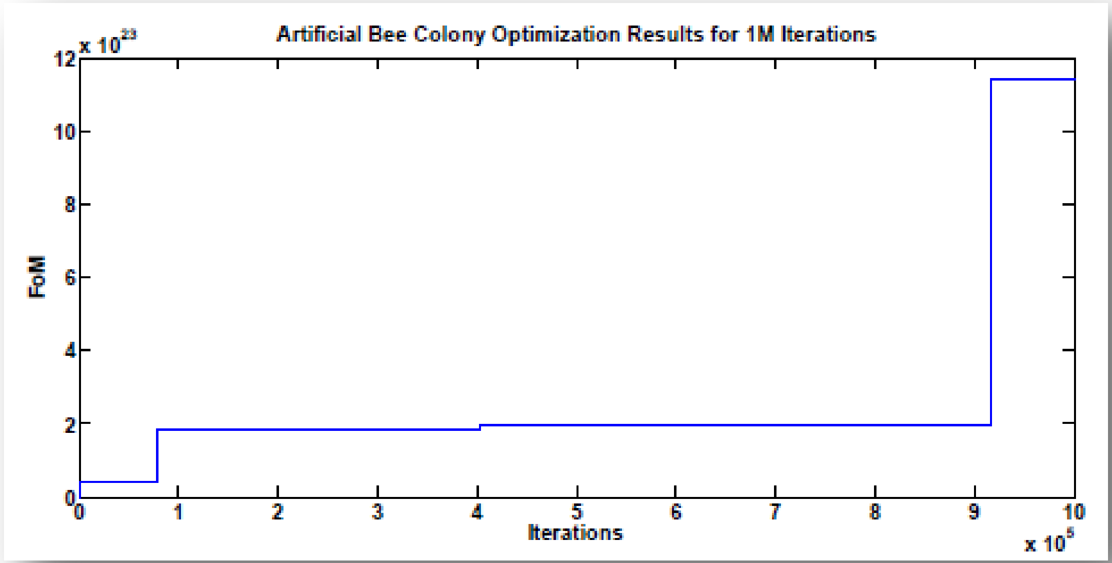
Generated R^2 and R^2_{adj} for various orders of the polynomial metamodel for settling time. Notice possible overfitting.

Metamodels of the PLL

- The number of coefficients corresponding to the order of the generated metamodel for settling time.
- This means that the model is over fitted, therefore for the metamodel that represents settling time, a polynomial order of 4 will be used.



Optimization of the PLL …



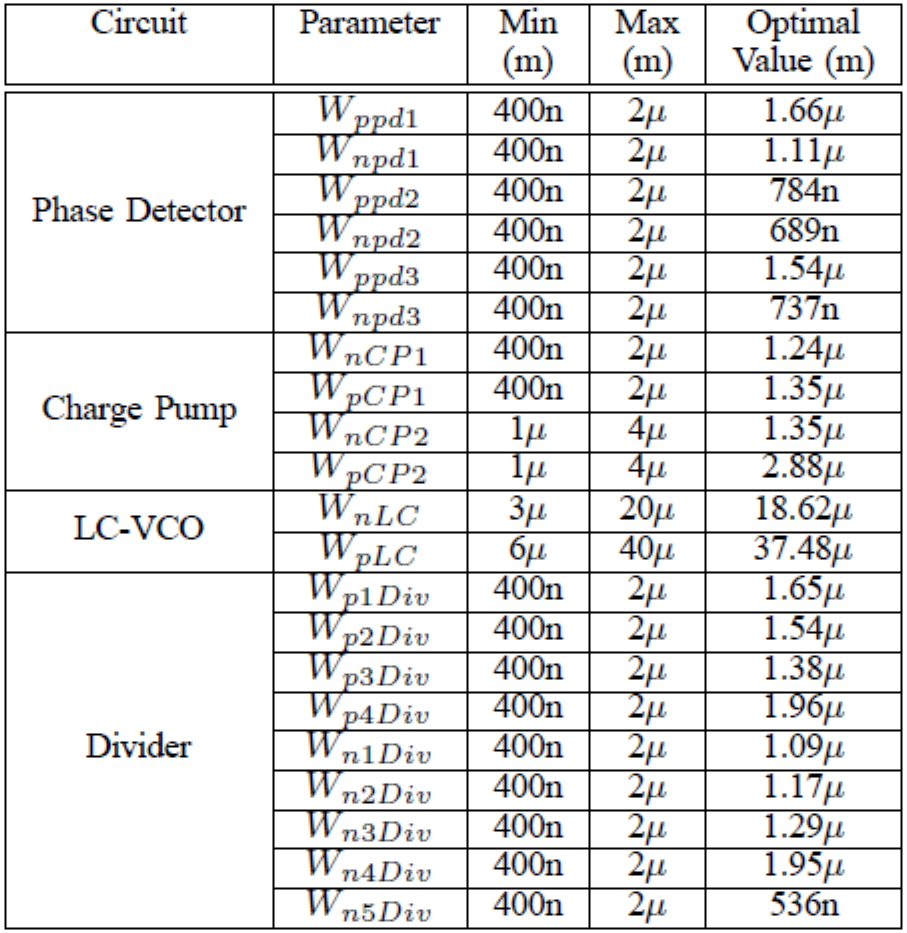
A Bee-Colony Optimization algorithm progression for the selected FoM.

Power and Jitter Results of the PLL

Metric	Before Optimization	After Optimization	Improvement
Power	9.29 mW	0.87 mW	90.6%
Jitter Vertical	$168.35\mu V$	3.28 nV	~100%
Jitter Horizontal	189 ps	180 ps	4.8%

Optimization of the PLL

PLL parameters with constraints and optimized values.



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

- An exhaustive search of the design space of 21 parameters with 10 intervals per parameter requires 10^{21} simulations.
- Time savings are enormous: ~10^{20}x.

Conclusions

- Paper investigated the use of metamodeling and an intelligent Bee Colony Optimization algorithm to speed up the design-space exploration for AMS circuits.
- A 180nm PLL, the circuit was parameterized with 21 parameters and optimized using the BCO algorithm.
- The final outcome of the design flow was 90% power savings and and average of 52% jitter minimization.
- Only 100 simulations are used to generate polynomial metamodels and BCO converged faster.
- An exhaustive search of the design space of 21 parameters with 10 intervals per parameter would require 10^{21} simulations. The time savings are enormous ($\approx 10^{20}\times$ simulation time).

Thank you !!!