Ultra-Fast Design Exploration of Nanoscale Circuits through Metamodeling

Saraju P. Mohanty, Oghenekarho Okobiah, and Geng Zheng
NanoSystem Design Laboratory (NSDL)
Dept. of Computer Science and Engineering
University of North Texas, Denton, TX 76203, USA.
Email: saraju.mohanty@unt.edu

Presented By
Oghenekarho Okobiah
Outline of the Talk

- Nanoscale Design Challenges
- The Proposed Ultra-Fast Solution
- Metamodel Types and Proposed Techniques
- Algorithms for Optimization over Metamodels
- Experiments Using Case Studies
- Conclusions and Future Research
Outline of the Talk

- **Nanoscale Design Challenges**
- The Proposed Ultra-Fast Solution
- Metamodel Types and Proposed Techniques
- Algorithms for Optimization over Metamodels
- Experiments Using Case Studies
- Conclusions and Future Research
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).

<table>
<thead>
<tr>
<th>DSP Core</th>
<th>Processor Core</th>
<th>Memory Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital ASIC Core</td>
<td>System Software</td>
<td>Application Software</td>
</tr>
<tr>
<td>Base Band Analog Circuitry Core</td>
<td>Mixed Signal Circuit Core</td>
<td>Radio Frequency Circuitry Core</td>
</tr>
</tbody>
</table>
One of the Key Issues: Time/Effort

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

Issues for AMS-SoC components:
- How fast can design space exploration be performed?
- How fast can layout generation and optimization be performed?
Standard Design Flow – Very Slow

- 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.

Flowchart:
- **Specs**
  - **Specs met?**
    - **NO**
    - **Logical Iteration loop (Automatic)**
    - **YES**
      - **Layout**
      - **DRC/LVS/RCLK Extraction**
      - **Physical—logical Iteration loop (Manual)**
      - **X iterations**
      - **Specs met?**
        - **NO**
        - **YES**
          - **Stop**
Outline of the Talk

- Nanoscale Design Challenges
- **The Proposed Ultra-Fast Solution**
- Metamodel Types and Proposed Techniques
- Algorithms for Optimization over Metamodels
- Experiments Using Case Studies
- Conclusions and Future Research
Automatic Optimization on Netlist
(Faster than manual flow; still slow)

- Automatic iteration over netlist improves design optimization.
- Still needs multiple simulations using analog simulator (SPICE).
- SPICE is slow.
Ultra-Fast Design Exploration Through Metamodeling

Baseline Design → Exhaustive Optimization → Optimized Design

The Actual Circuit (Netlist) Optimization -- Slow Approach

Baseline Design → Accurate Metamodeling → Fast Optimization → Optimized Design

The Metamodel-Based Approach -- Ultra-Fast Approach
Two Tier Speed Up

Traditional – Slow Approach

Baseline Mixed-Signal Circuit
Layout-Aware Netlist

Metamodels of Baseline Mixed-Signal Circuit

9000x Speedup

Optimization over Metamodels
300x Speedup

Technology Constraints
Specification Constraints

Parameter i
Parameter j
Proposed Flow: Key Perspective

- Novel design and optimization methodology that will produce robust AMS-SoC components using **ultra-fast automatic iterations over metamodels** (instead of netlist) and two manual layout steps.

- The methodology easily accommodates multidimensional challenges, reduces design cycle time, improves circuit yield, and reduces chip cost.
Metamodel-Based Design Flow

Input Specifications of the Mixed-Signal IC

- Create Logical Design
- Mixed-Signal Design Schematic

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

Specifications met?

Yes

Done

No

- 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 Metamodel Generation

Create Metamodels of Figures-of-Merits of the Mixed-Signal IC

- Metamodels of the FoMs of the Mixed-Signal IC

Use an Algorithm to Perform Optimization Over Polynomial Metamodels

- Optimal Physical-Design Variables

Specifications met?

Yes

Optimized Mixed-Signal IC Layout

Create New Layout of the Mixed-Signal IC

Perform DRC/LVS/RCLK Extraction

Done
Metamodeling vs. Macromodeling

- **Macromodeling**
  - Simplified version of the circuit.
  - Used in the same simulation tool.
  - Hard to create.

- **Metamodelling**
  - Mathematical representation of output.
  - Based on prediction equation or algorithm.
  - Language and tool independent.
  - Reusable for different specifications.
  - Can be applied using non-EAD tools like MATLAB.
Outline of the Talk

- Nanoscale Design Challenges
- The Proposed Ultra-Fast Solution
- **Metamodel Types and Proposed Techniques**
- Algorithms for Optimization over Metamodels
- Experiments Using Case Studies
- Conclusions and Future Research
Metamodels: Selected Types

Nanoscale-CMOS Circuit Metamodels

Polynomial
- Regular Polynomial
- Piece-wise Polynomial

Nonpolynomial
- Neural-Networks
- Kriging Methods
Metamodels: Polynomial Example

Actual Circuit (SPICE netlist) of AMS-SoC Components

Statistical Sampling

Polynomial Function Fitting

\[ f(W_n, W_p) = 7.94 \times 10^9 + 1.1 \times 10^{16} W_n + 1.28 \times 10^{15} W_p. \]
Metamodelling – Key Points

- **Accuracy** -- Capability of predicting the system response over the design space.
- **Efficiency** -- Computational effort required for constructing the metamodel.
- **Transparency** -- Capability of providing the information concerning contributions and variations of design variables and correlation among the variables.
- **Simplicity** -- Simple methods should require less user input and be easily adapted to different problem.
Metamodels: Performance Analysis

- **Root-Mean Square Error (RMSE):** Represents departure of metamodel from real-simulation (golden). Smaller RMSE means more accurate:

\[
RMSE = \sqrt{\left(\frac{1}{N}\right) \sum_{k=1}^{N} \left(FoM(x_k) - \overline{FoM}(x_k)\right)^2}
\]

- **Relative Average Absolute Error (RAAE):** Smaller RAAE means more accurate metamodel:

\[
RAAE = \left(\frac{\sum_{k=1}^{N} |FoM(x_k) - \overline{FoM}(x_k)|}{N \times \text{Standard Deviation}}\right)
\]

- **R-Square:** Larger R-square means more accurate metamodel:

\[
R^2 = \left(1 - \frac{MSE}{Variance}\right)
\]
Metamodel Generation Flow

Parasitic-Aware Parameterized Netlist of Mixed-Signal Components

Perform Statistical Sampling of Mixed-Signal Component Design Space

Perform Polynomial, Piecewise Polynomial Fitting

Explore Different Order and Type Polynomials

Perform Statistical Analysis of the Metamodels

Ranking of Metamodels

- Different flow is used for nonpolynomial metamodel generation.
Sampling Techniques: 45nm Ring Oscillator Circuit (5000 points)

Monte Carlo

MLHS

LHS

DOE
Sampling Comparison: RO / LC-VCO

RMSE Comparison for Ring Oscillator

RMSE (MHz)

Number of Simulations

500 1000 1500 2000 2500 3000 3500 4000 4500 5000

RMSE Comparison for LC-VCO

RMSE (MHz)

Number of Samples

0 200 400 600 800 1000 1200 1400 1600 1800 2000

MC
LHS
MLHS

06/26/2012

Mohanty 22
Polynomial Metamodels

- The generated sample data can be fitted in many ways to generate a metamodel.
- The choice of fitting algorithm can affect the accuracy of the metamodel.
- A simple metamodel has the following form:

\[ y = \sum_{i,j=0}^{k} (\alpha_{ij} \times x_1^i \times x_2^j) \]

- \(y\) is the response being modeled (e.g. frequency), \(x = [W_n, W_\rho]\) is the vector of variables and \(\alpha_{ij}\) are the coefficients.
### Metamodel: Polynomial Comparison

<table>
<thead>
<tr>
<th>Case Study Circuits</th>
<th>Polynomial Order</th>
<th>$\mu$ error (in MHz)</th>
<th>$\sigma$ error (in MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ring Oscillator</td>
<td>1</td>
<td>571.0</td>
<td>286.7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>195.4</td>
<td>78.1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>37.2</td>
<td>18.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>20.0</td>
<td>10.7</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>17.1</td>
<td>9.6</td>
</tr>
<tr>
<td>45nm CMOS</td>
<td>1</td>
<td>42.3</td>
<td>40.1</td>
</tr>
<tr>
<td>Target $f$: 10GHz</td>
<td>2</td>
<td>39.4</td>
<td>37.8</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>35.4</td>
<td>33.9</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>30.5</td>
<td>29.3</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>26.5</td>
<td>25.2</td>
</tr>
<tr>
<td>LC-VCO</td>
<td>1</td>
<td>2.38 $\times 10^9$ - 3.49 $\times 10^{12}$ $W_n$ - 6.66 $\times 10^{12}$ $W_p$.</td>
<td></td>
</tr>
</tbody>
</table>

**Ring oscillator – Order 1**

$$f(W_n, W_p) = 7.94 \times 10^9 + 1.1 \times 10^{16} W_n + 1.28 \times 10^{15} W_p.$$
Feed-forward dual layer NNs (FFDL) are considered.

FFDL network created for each FoM:
- Nonlinear hidden layer functions are considered each varying hidden neurons 1-20:
  \[ b_j(v_j) = \tanh(\lambda v_j) \]
Metamodel Comparison: Polynomial Vs Nonpolynomial

- Nonpolynomial (Neural Network) is more suitable for large circuits.

180nm CMOS PLL with Target Specs: \( f = 2.7\text{GHz}, P = 3.9\text{mW}, 8.5\mu\text{s}. \)

<table>
<thead>
<tr>
<th>Figures-of-Merits (FoM)</th>
<th>Polynomial</th>
<th>Nonpolynomial (Neural Network)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of Coefficients</td>
<td>RMSE</td>
</tr>
<tr>
<td>Frequency</td>
<td>48</td>
<td>77.96 MHz</td>
</tr>
<tr>
<td>Power</td>
<td>50</td>
<td>2.6mW</td>
</tr>
<tr>
<td>Locking Time</td>
<td>56</td>
<td>1.9\mu s</td>
</tr>
</tbody>
</table>

- 56% increase in accuracy over polynomial metamodels.
- On average 3.2% error over golden design surface.
Outline of the Talk

- Nanoscale Design Challenges
- The Proposed Ultra-Fast Solution
- Metamodel Types and Proposed Techniques
- **Algorithms for Optimization over Metamodels**
- Experiments Using Case Studies
- Conclusions and Future Research
Selected Algorithms for Optimization over Metamodels

Algorithms Applied Over Metamodels

Traditional Heuristics
- Simulated Annealing
- Tabu Search

Intelligent Algorithms
- Artificial Bee Colony (ABC)
- Ant Colony
Exhaustive Search: 45nm RO

- Searches over two parameter space.
- Parameters incremented over specified steps.
DOE Assisted Tabu Search: 45nm RO

- Search space is recursively divided into rectangles and each time the rectangle with superior result is selected.
Comparison of the Running Time of Heuristic Algorithms: 45nm RO

- **Optimization without metamodels:** the tabu search optimization is faster by ~1000× than the exhaustive search and ~4× faster than the simulated annealing optimization.

- **Optimization with metamodels:** the simulated annealing optimization is faster by ~1000× than the exhaustive search and ~6× faster than the tabu search optimization.
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 scout bees.
   
   4) Best food source determined so far is recorded.

3. **While** (requirements are met)

   A food source → a solution; A position of a food source → a design variable set; Nectar amount → Quality of a solution; Number of worker bees → number of quality solutions.
Outline of the Talk

- Nanoscale Design Challenges
- The Proposed Ultra-Fast Solution
- Metamodel Types and Proposed Techniques
- Algorithms for Optimization over Metamodels
- **Experiments Using Case Studies**
- Conclusions and Future Research
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.
PLL: Polynomial Metamodels ...

- 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 $R^2_{\text{adj}}$ for various orders of the polynomial metamodel for settling time. Notice possible overfitting.
PLL: Polynomial Metamodels ...

- 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.
The Artificial Bee-Colony (ABC) Optimization algorithm progression for the selected FoM.

Power and Jitter Results of the PLL

<table>
<thead>
<tr>
<th>Metric</th>
<th>Before Optimization</th>
<th>After Optimization</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power</td>
<td>9.29 mW</td>
<td>0.87 mW</td>
<td>90.6%</td>
</tr>
<tr>
<td>Jitter Vertical</td>
<td>168.35 μV</td>
<td>3.28 nV</td>
<td>~100%</td>
</tr>
<tr>
<td>Jitter Horizontal</td>
<td>189 ps</td>
<td>180 ps</td>
<td>4.8%</td>
</tr>
</tbody>
</table>
PLL parameters with constraints and optimized values.

- An exhaustive search of the design space of 21 parameters with 10 intervals per parameter requires $10^{21}$ simulations.
- $10^{21}$ SPICE simulations is slow; 10min per one.
- $10^{21}$ simulations using polynomial metamodels is fast.
- Time savings: $\approx 10^{20}$x SPICE simulation time.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Parameter</th>
<th>Min (m)</th>
<th>Max (m)</th>
<th>Optimal Value (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase Detector</td>
<td>$W_{pdp1}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.66$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{ndp1}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.11$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{pdp2}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>784n</td>
</tr>
<tr>
<td></td>
<td>$W_{ndp2}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>689n</td>
</tr>
<tr>
<td></td>
<td>$W_{pdp3}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.54$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{ndp3}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>737n</td>
</tr>
<tr>
<td>Charge Pump</td>
<td>$W_{nCP1}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.24$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{pCP1}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.35$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{nCP2}$</td>
<td>1$\mu$</td>
<td>4$\mu$</td>
<td>1.35$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{pCP2}$</td>
<td>1$\mu$</td>
<td>4$\mu$</td>
<td>2.88$\mu$</td>
</tr>
<tr>
<td>LC-VCO</td>
<td>$W_{nLC}$</td>
<td>3$\mu$</td>
<td>20$\mu$</td>
<td>18.62$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{pLC}$</td>
<td>6$\mu$</td>
<td>40$\mu$</td>
<td>37.48$\mu$</td>
</tr>
<tr>
<td>Divider</td>
<td>$W_{p1Div}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.65$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{p2Div}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.54$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{p3Div}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.38$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{p4Div}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.96$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{n1Div}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.09$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{n2Div}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.17$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{n3Div}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.29$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{n4Div}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>1.95$\mu$</td>
</tr>
<tr>
<td></td>
<td>$W_{n5Div}$</td>
<td>400n</td>
<td>2$\mu$</td>
<td>536n</td>
</tr>
</tbody>
</table>
PLL: ABC Optimization: Poly. Vs NN

- Figure-of-Merit used for optimization objective function of PLL: 
  \[ FoM = \left( \frac{1}{\text{Power} \times \text{Locking Time}} \right) \].

Artificial Bee Colony Optimization Using Polynomial and ANN Metamodels
## Optimization Results

<table>
<thead>
<tr>
<th>FoM</th>
<th>Poly. Metamodel</th>
<th>ANN Metamodel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Power</td>
<td>3.9 mW</td>
<td>3.9 mW</td>
</tr>
<tr>
<td>Frequency</td>
<td>2.6909 GHz</td>
<td>2.7026 GHz</td>
</tr>
</tbody>
</table>

## Optimization Time Comparison

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Circuit Netlist</th>
<th>Poly. Metamodel</th>
<th>ANN Metamodel</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC (100 iterations)</td>
<td>#bees(20) * 5 min * 100 iteration = 10,000 minutes = 7 days (worst case)</td>
<td>5 mins</td>
<td>0.12 mins</td>
</tr>
<tr>
<td>Metamodel Generation</td>
<td>0</td>
<td>11 hours for LHS + 1 min creation</td>
<td>11 hours for LHS + 10mins training and verification.</td>
</tr>
</tbody>
</table>
Outline of the Talk

- Nanoscale Design Challenges
- The Proposed Ultra-Fast Solution
- Metamodel Types and Proposed Techniques
- Algorithms for Optimization over Metamodels
- Experiments Using Case Studies
- Conclusions and Future Research
Related Prior Research

Fast Design Exploration

Neural Networks
Xia 2009
Wolfe 2003
Xu 2003

Intelligent Algorithms
Delican 2010
Thakker 2009
Fernando 2008

Macromodeling
Basu 2009
Ding 2006
Agarwal 2005

Metamodeling
Samanta 2010
Lamecki 2008
Wong 2006

Actual SPICE netlist

Simplified SPICE

Functions
Conclusions …

- Polynomial/nonpolynomial metamodels are explored.
- Use of metamodels and optimization algorithm speed up the design-space exploration for AMS circuits.
- LHS was identified as an accurate sampling method.
- Polynomial metamodels are easier create but can be applied for small circuits.
- 56% increase in accuracy is observed using feed forward NN over polynomial metamodels.
- On average 3.2% error is observed using NN.
Conclusions

- As a case study, a 180nm PLL, the circuit was parameterized with 21 parameters and optimized using the ABC algorithm.
- The final outcome of the design flow was 90% power savings and an average of 52% jitter minimization.
- Only 100 simulations are used to generate the accurate metamodels and ABC 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 \text{SPICE simulation time}).
Our Selected Publication on this Research


Future Research

- Capturing statistical process variations using metamodels
- Kriging metamodelling
  - Effective handle correlations
  - Accurately model process variations
- Integration in HDLs
  - Used for accurate behavioral simulations
- Application to MEMS/NEMS
  - Unified simulation and design exploration of heterogeneous components
Thank you !!!