Particle Swarm Optimization over Non-Polynomial Metamodels for Fast Process Variation Resilient Design of Nano-CMOS PLL

Oleg Garitselov, Saraju P. Mohanty, Elias Kougianos, Geng Zheng
NanoSystem Design Laboratory (http://nsdl.cse.unt.edu), Dept. of Computer Science and Engineering
University of North Texas, USA. Email-ID: saraju.mohanty@unt.edu

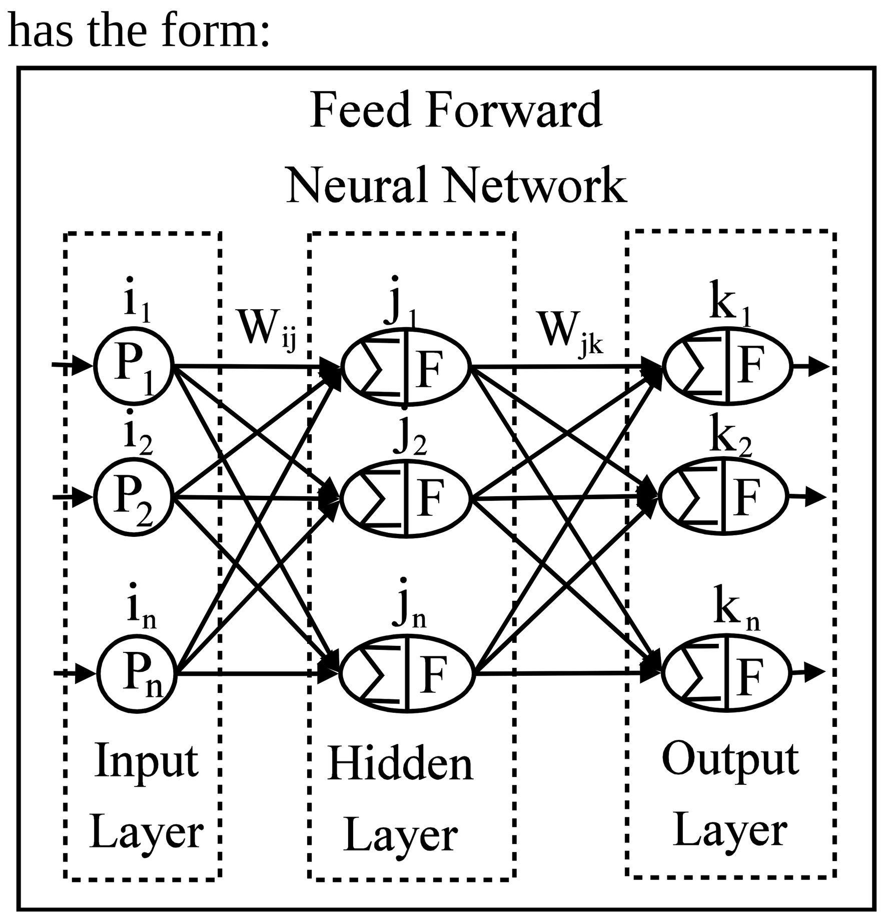
Abstract

A automated top-down design flow to achieve physical design of an AMS-SoC has always been very difficult. The design efforts have been further increased when the silicon is due to nanoscale CMOS. The various nanoscale effects, particularly, the process variation effects have profound effects on the performance of the performance of silicon versus the layout design. In this paper metamodels (aka surrogated models) and particle swarm optimization (PSO) have been combined in an automated physical design flow for fast design exploration of AMS-SoC. The neural network based non-polynomial metamodels that handles large number of design parameters are used to predict the statistical process variation effects instead of the exhaustive Monte Carlo simulations over the circuit netlist. The statistical analysis over non-polynomial metamodels were as very fast while the error was only 0.7%. The PSO algorithm is used for optimization of the AMS-SoC components using their non-polynomial metamodels instead of the actual circuit. The PSO algorithm followed a two step approach: local and global. The physical design phase-locked loop (PLL) is considered as a case study circuit. The proposed design flow is approximately 5 times faster while the error is under 2% compared to the Monte Carlo analysis.

Introduction

A metamodel is a mathematical formula that represents the circuit's behavior within a given range of parameters and is derived from sampling points.

The neural network metamodel considered has the form:



A metamodel is created on a full RCLK (resistance, capacitance, self and mutual inductance) parasitic extracted netlist, for each figure of merit.

This study is attempting to answer two main questions for mixed signal circuits:

•How accurately can metamodels predict process variation behavior?

•Can metamodels be used for optimization and account for process variation?

Case Study Circuit

The proposed approach has been used on a phase locked loop (PLL) designed in 180 nm technology. The figure below shows PLL structure.

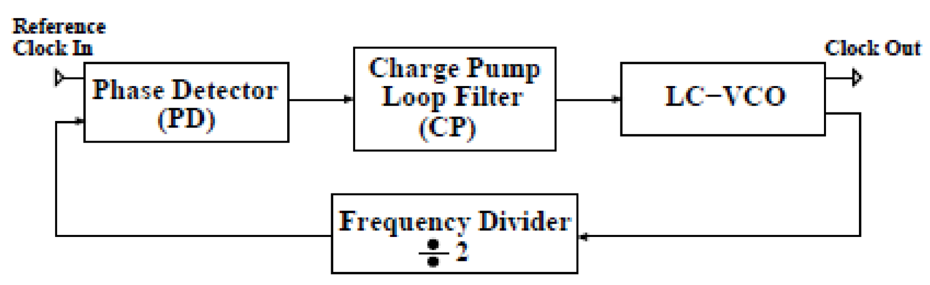


Figure 1: Block diagram of a phase locked loop.

Schematic and then physical layout is created for the circuit.

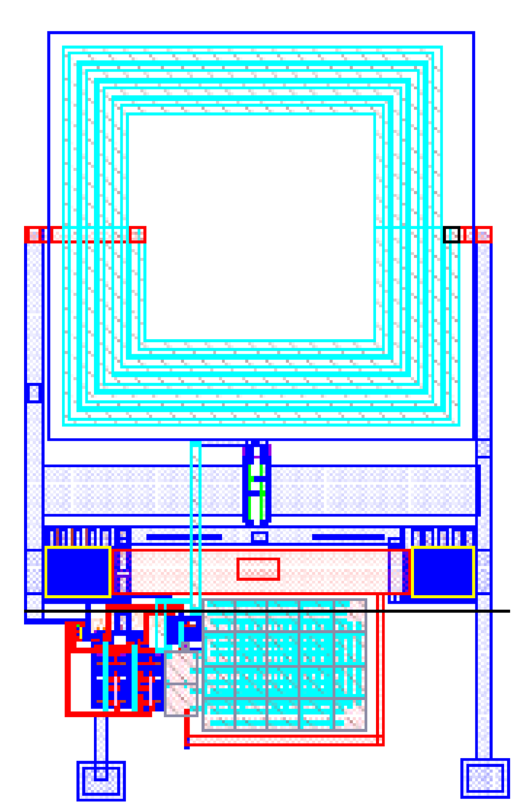


Figure 4: Physical Layout of the optimized PLL.

Extracted parasitic netlist is characterized with a wide range for 35 parameters. Parameters include device sizing, VDD variation, and process variation parameters s.a. threshold voltage and oxide thickness.

Table 1: Parameter Ranges and Optimization Results For WiiMAX and MMDS Specifications

Circuit	Component	Parameter Name	min	max
Phase Detector	DFF1 PMOS	W_{ppd1}	0.4 μm	2 μm
	DFF1 NMOS	W_{npd1}	0.4 μm	2 μm
	DFF1 Length	L_{npd1}	180 nm	200 nm
	DFF2 PMOS	W_{ppd2}	0.4 μm	2 μm
	DFF2 NMOS	W_{npd2}	0.4 μm	2 μm
	DFF2 Length	L_{npd2}	180 nm	200 nm
	AND PMOS	W_{ppd3}	0.4 μm	2 μm
	AND NMOS	W_{npd3}	0.4 μm	2 μm
	AND Length	L_{npd3}	180 nm	200 nm
Charge Pump	M3, M4	W_{pCP1}	0.4 μm	2 μm
	M5, M6	W_{nCP1}	0.4 μm	2 μm
	M1, M2	W_{pCP2}	4 μm	20 μm
	M7, M8, M9	W_{nCP2}	2 μm	20 μm
	Length NMOS	L_{nCP}	180 nm	200 nm
	Length PMOS	L_{pCP}	180 nm	200 nm
LC-VCO	NM1, NM2	W_{nLC}	3 μm	20 μm
	PM1, PM2	W_{pLC}	6 μm	40 μm
	Length PMOS	L_{pLCVCO}	180 nm	200 nm
	Length NMOS	L_{nLCVCO}	180 nm	200 nm
Dividor	M5	W_{n1Div}	0.4 μm	2 μm
	M6	W_{n2Div}	0.4 μm	2 μm
	M7	W_{n3Div}	0.4 μm	2 μm
	M8	W_{n4Div}	0.4 μm	2 μmn
	M9	W_{n5Div}	0.4 μm	2 μm
	M1	W_{p1Div}	0.4 μm	2 μm
	M2	W_{p2Div}	0.4 μm	2 μm
	M3	W_{p3Div}	0.4 μm	2 μm
	M4	W_{p4Div}	0.4 μm	2 μm
	Length PMOS	L_{pDIV}	180 nm	200 nm
	Length NMOS	L_{nDIV}	180 nm	200 nm
Global	Oxide Thickness NMOS	Tox_n	2 nm	5 nm
	Oxide Thickness PMOS	Tox_p	2 nm	5 nm
	Threshold Voltage NMOS	VTH_N	0.08 V	0.88 V
	Threshold Voltage PMOS	VTH_P	0.03 V	0.83 V
	Supply voltage	Vdd	1 V	1.4 V

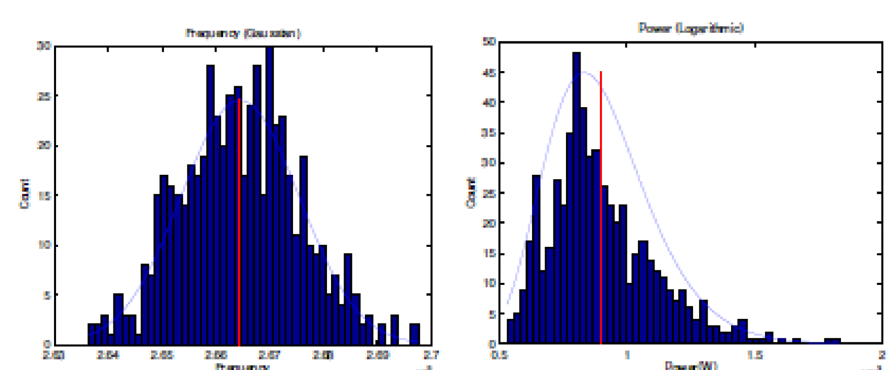
200 training samples are used to create the neural model. 60 samples are used for verification.

Table 2: Before Optimization: Statistical Figures of Merits of the PLL.

	Circuit Monte Carlo		Neural Network Monte Carlo		Error	
	Mean (μ)	Standard Deviation (σ)	Mean (μ)	Standard Deviation (σ)	Mean (μ)	Standard Deviation (σ)
Frequency	2.66 GHz	10.95 MHz	2.66 GHz	10.96 MHz	0.0%	0.11%
Power	0.90 mW	0.21 mW	0.90 mW	0.21 mW	0.14%	1.3%
Locking Time	3.24 μs	1.07 μs	3.22 μs	0.99 μs	0.7%	6.93%
Horizontal Jitter	2.79 ps	1.32 ps	2.80 ps	1.32 ps	0.12%	0.5%
Vertical Jitter	0.41 mV	0.17 mV	0.41 mV	0.15 mV	0.53%	10.02%

Neural Model Verification

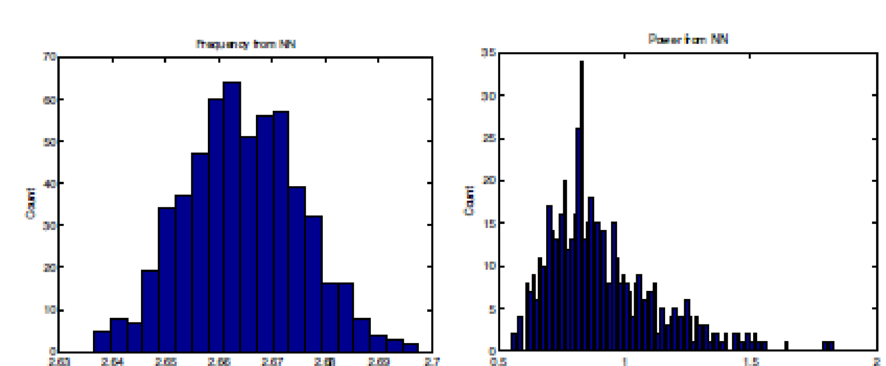
After successful training of the neural model thethe accuracy of neural model for process variation is checked by conducting Monte Carlo analysis of 1000 points on physical netlist and neural model.



((a)) Frequency ((b)) Power

((c)) Locking Time ((d)) Horizontal Jitter

Figure 2: Statistical Analysis of the FoM of PLL using *Actual Circuit (netlist).*



((a)) Frequency ((b)) Power

((c)) Locking Time ((d)) Horizontal Jitter

Figure 3: Statistical Analysis of the FoM of PLL using *Neural Network based Non-Polynomial Metamodels.*

Particle Swarm Optimization

Particle Swarm optimization algorithm is used on the neural model to find optimal values that are process variation resilient.

Algorithm 1 Particle Swarm Optimization (PSO)

```
Set N - number of particles
Start at a random location with uniform distribution
Get current position x_i and use it initially for best particle position f(p_i) and f(g) = min(p_i)
v_i U(min_p_i, max_p_i)
Initialize iter=0
Initialize weight for swarm effect ϱ_p
Initialize weight for swarm effect ϱ_p
Initialize weight for velocity effect (acceleration/inertia) w
while iter<max_iterations do
    for each i do
        v_i = ωv_i + ϱ_p τ_p (p_i − x_i) + ϱ_g τ_g (g − x_i)
        x_i ← x_i + v_i
        if f(x_i) < f(p_i) then
            update position: p_i ← x_i
            if f(p_i) < f(g) then
                g ← p_i
            end if
        end if
    end for
end while
```

Algorithm 2 Cost Function $f(p_i)$ Calculation

```
Receive parameters
Conduct Monte Carlo 1000 points
Calculate freq_μ,freq_σ=frequency(μ,σ)
if freq_μ+freq_σ<freq_constraint then
    Calculate power_μ,power_σ=power(μ,σ)
    Calculate lockingtime_μ,lockingtime_σ=lockingtime(μ,σ)
    Calculate horjitter_μ,horjitter_σ=horjitter(μ,σ)
    Normalize μ + k ∗ σ for all values
    Return FoM=sum(normalized values)
end if
```

Conclusion

PLL circuit is characterized for Frequency, Power consumption, Locking time and horizontal jitter of the output signal.

Optimization is conducted for:

a) $\mu+\sigma$

b) $\mu+3\sigma$

Optimization results are shown in Table 3.

An error of under 2% has been observed in the models for process variation analysis for and standard deviation. Mc analysis for 1000 simulation points for PLL netlist took approximately 5 days in comparison to 1 day for 200 neural network training points. The speed up of approx. 5X is observed for using NN for optimization.

Table 3: After Optimization: Statistical Figures of Merits of the PLL.

	$\mu+\sigma$ Optimization		$\mu+3\sigma$ Optimization	
	Mean (μ)	Standard Deviation (σ)	Mean (μ)	Standard Deviation (σ)
Frequency	2.75 GHz	28.64 MHz	2.74 GHz	29.14 MHz
Power	0.99 mW	0.28 mW	0.98 mW	0.27 mW
Locking Time	4.69 μs	1.15 μs	4.61 μs	1.13 μs
Horizontal Jitter	5.82 ps	3.42 ps	5.97 ps	3.34 ps

This research is supported in part by NSF awards CNS-0854182 and DUE-0942629, and SRC contract P10883.