Fast-Accurate Non-Polynomial Metamodeling for nano-CMOS PLL Design Optimization

O. M. Garitselov[1], S. P. Mohanty[2] and E. Kougianos[3]

NanoSystem Design Laboratory (NSDL, http://nsdl.cse.unt.edu)
University of North Texas, Denton, TX 76203, USA.[1,2,3]

Email: omg0006@my.unt.edu [1], saraju.mohanty@unt.edu[2] and eliask@unt.edu[3]

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

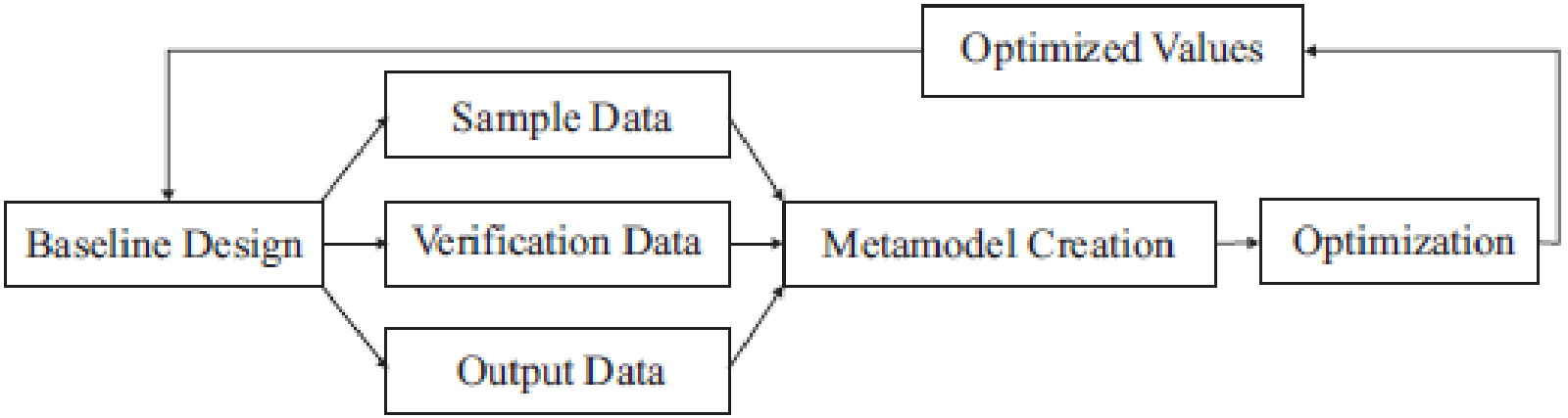
Outline of the Talk

- Introduction
- Proposed Design Flow
- Phase Locked Loop Circuit
- Polynomial and Neural Network Metamodeling
- Bee Colony Optimization Algorithm
- Results and Comparison
- Conclusions and Future Research Directions

Introduction

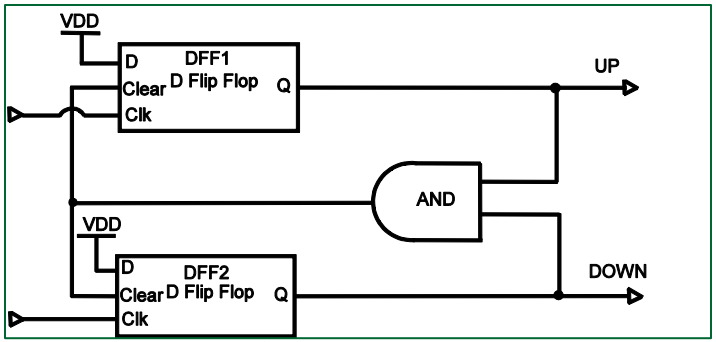
- Goal: Reduce the complexity of computations for analog circuits to include parasitics.
- Physical layout and simulation analysis is very time consuming processes in the design flow.
- Metamodels use mathematical formulas that represent circuit behavior within a given range using small amount of sampling points.
- This paper targets sampling techniques which are technology independent and the amount that is needed to create an accurate metamodel

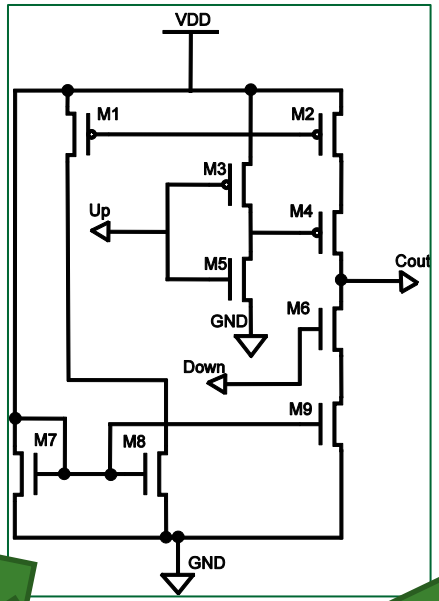
Design Flow

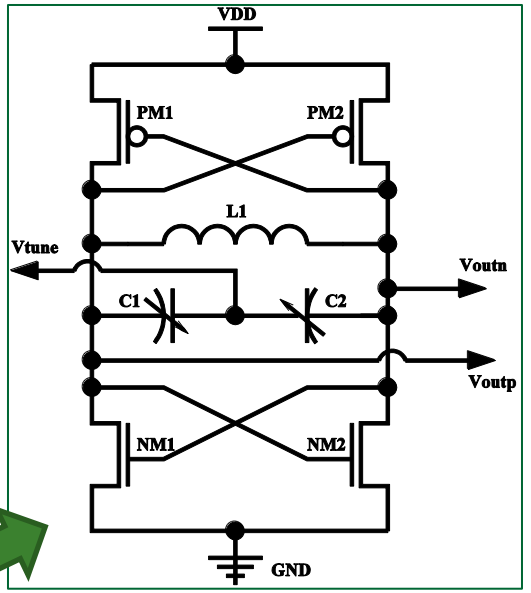


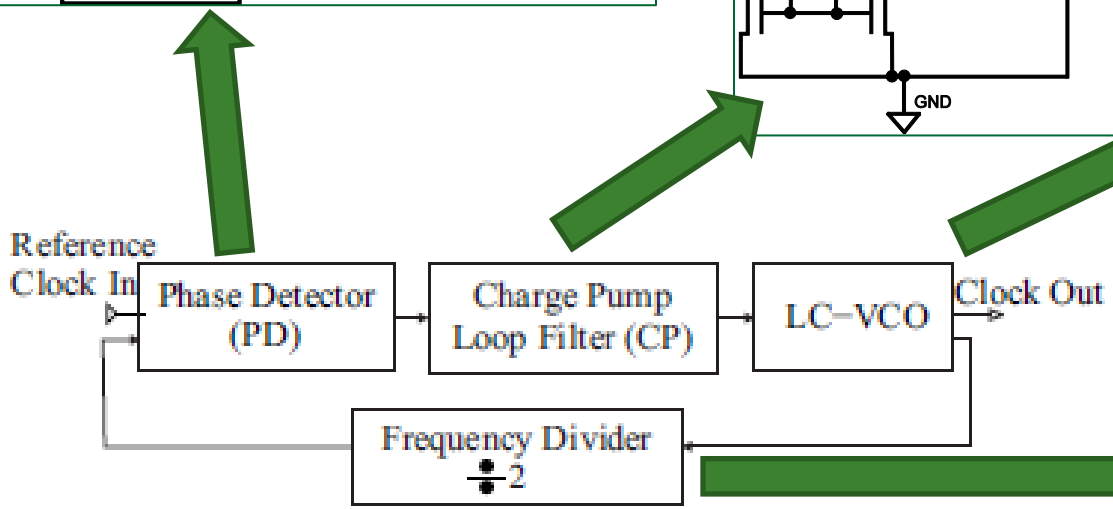
- Verification data set is 30% < Sample (training) data set.
- Most of the work is done outside of SPICE simulator.

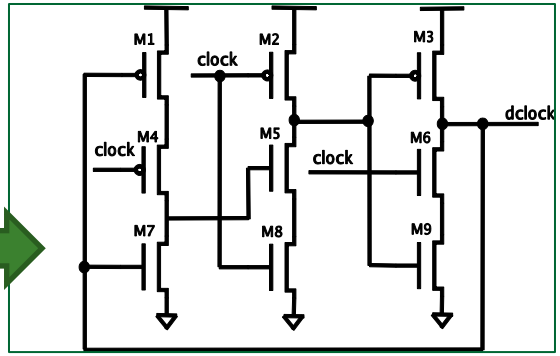
Phase Locked Loop



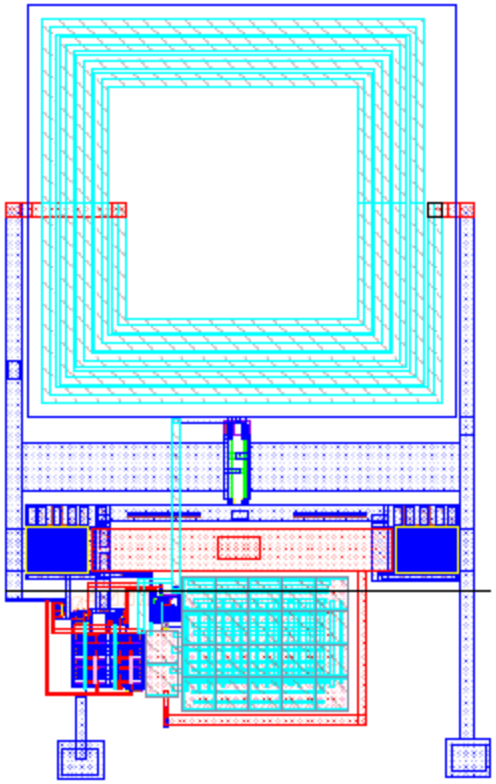




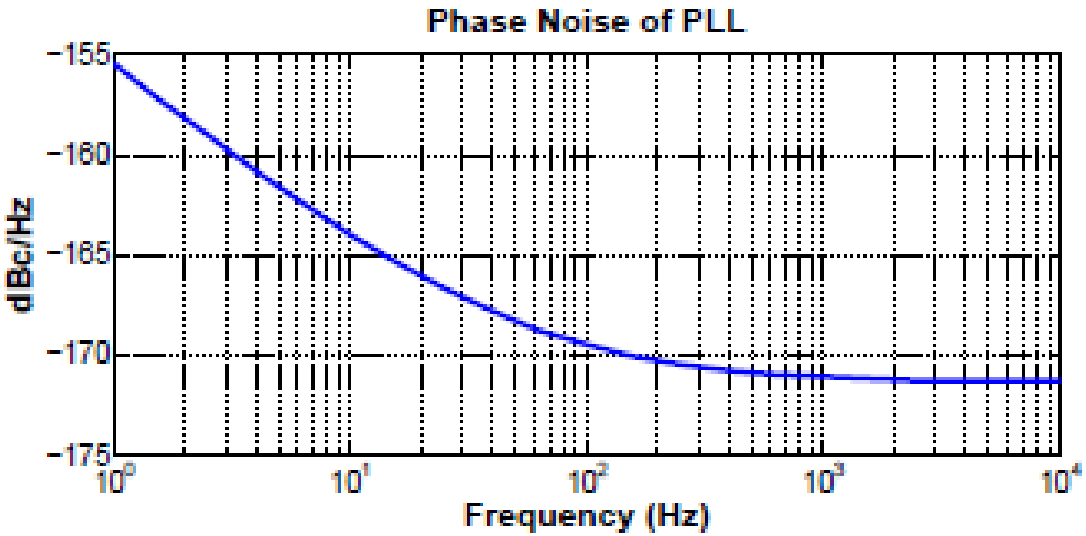




Phase Locked Loop



PLL physical layout



Parameters Considered in PLL

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

Polynomial Metamodels

- 21 design parameters used.
- 100/30 samples points used.
- Polynomial models are created using the stepwise regression to minimize the amount of coefficients.
- Order 1-6 models are picked for each FoM.
- Best models from each FoM, based on RMSE and r^2 are picked for both training and verification sets.

Polynomial Regression

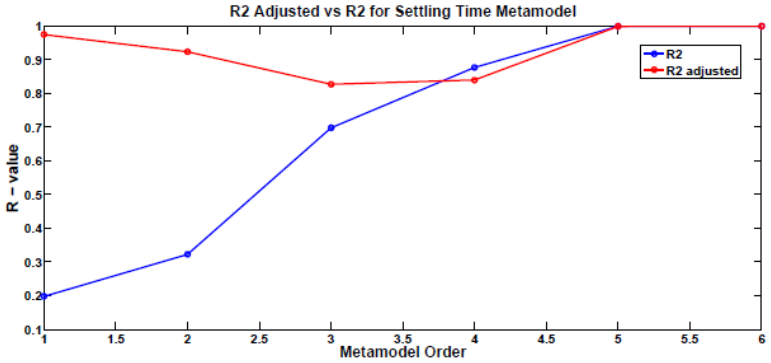


Fig. 7. Generated R^2 and R^2_{adj} for every order of the polynomial metamodel for settling time

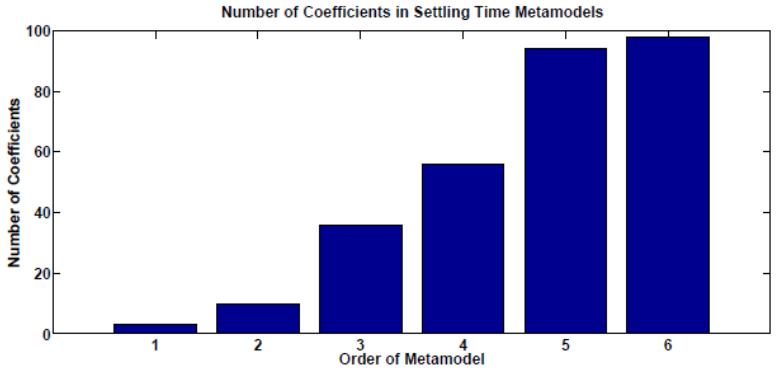


Fig. 8. Number of coefficients corresponding to the order of the generated metamodel for settling time

Neural Network Models

- 100/30 sample points used.
- Feed-forward dual layer NNs (FFDL) and radial NNs are considered in this work.
- Multiple FFDL networks for each FoM:
  - Two types of non-linear hidden layer functions are considered each varying hidden neurons 1-20:
    - A) $b_j(v_j) = \left(\frac{1}{1+e^{-\lambda v_j}}\right)$
    - B) $b_j(v_j) = \tanh(\lambda v_j)$

Neural Network Models

TABLE I
FREQUENCY NON-POLYNOMIAL METAMODEL COMPARISON OF THE PLL.

Function	Data Filtering	R^2-test	R^2- verification	RMSE	Neurons
logsig→purelin	none	0.802	0.723	52.74 MHz	4
tansig→purelin	none	0.839	0.713	51.24 MHz	3
radial→purelin	none	0.020	0.490	81.51 MHz	
logsig→purelin	minmax	0.917	0.664	48.89 MHz	9
tansig→purelin	minmax	0.855	0.699	53.65 MHz	1
radial→purelin	minmax	0.844	0.712	50.88 MHz	
logsig→purelin	meanstd	0.843	0.733	53.60 MHz	1
tansig→purelin	meanstd	0.793	0.762	51.64 MHz	5
radial→purelin	meanstd	0.848	0.749	48.97 MHz	
	Data Filtering	R^2	Order	RMSE	Number of Coefficients
polynomial	none	0.930	4	77.96 MHz	48

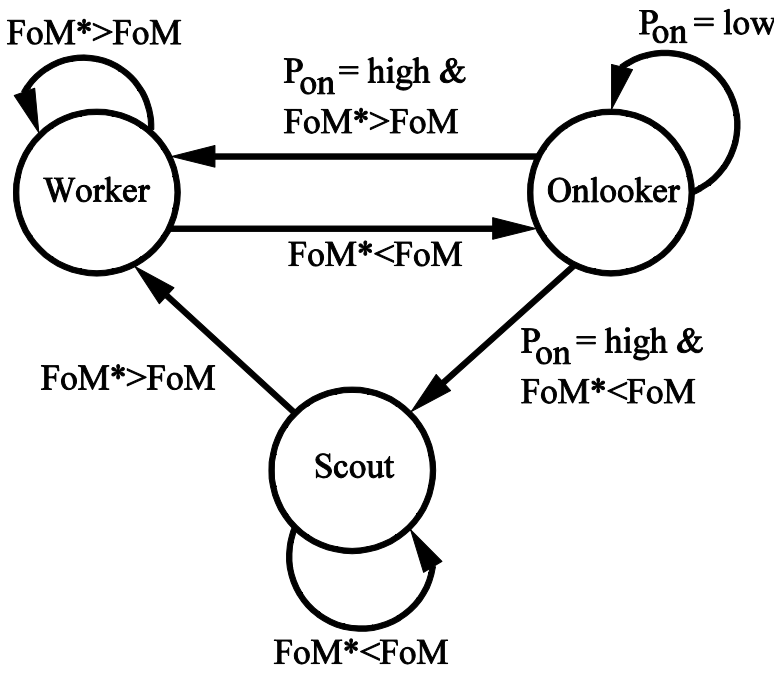
TABLE II
LOCKING TIME NON-POLYNOMIAL METAMODEL COMPARISON OF THE PLL.

Function	Data Filtering	R^2-Test	R^2-Verification	RMSE	Neurons
logsig→purelin	none	0.828	0.873	1.30 μs	1
tansig→purelin	none	0.850	0.723	1.44 μs	9
radial→purelin	none	0.078	0.830	2.26 μs	
logsig→purelin	minmax	0.826	0.870	1.29 μs	1
tansig→purelin	minmax	0.839	0.942	1.12 μs	10
radial→purelin	minmax	0.931	0.508	1.65 μs	
logsig→purelin	meanstd	0.826	0.906	1.22 μs	2
tansig→purelin	meanstd	0.737	0.939	1.12 μs	3
radial→purelin	meanstd	0.963	0.691	1.23 μs	
	Data Filtering	R^2	Order	RMSE	Number of Coefficients
polynomial	none	0.877	4	1.91 μs	56

TABLE III
POWER MODELS COMPARISON

Function	Data Filtering	R^2-test	R^2- verification	RMSE	Neurons
logsig→purelin	none	0.235	0.314	3.2 mW	10
tansig→purelin	none	0.217	0.233	2.8 mW	1
radial→purelin	none	0.533	0.582	2.6 mW	
logsig→purelin	minmax	0.995	0.990	325.30 μW	1
tansig→purelin	minmax	0.996	0.991	316.20 μW	1
radial→purelin	minmax	0.996	0.992	300.55 μW	
logsig→purelin	meanstd	0.995	0.992	300.63 μW	1
tansig→purelin	meanstd	0.996	0.993	281.29 μW	1
radial→purelin	meanstd	0.996	0.992	290.26 μW	
	Data Filtering	R^2	Order	RMSE	Number of Coefficients
polynomial	none	0.998	4	2.66 mW	50

Bee Colony Optimization



State diagram for bee transition

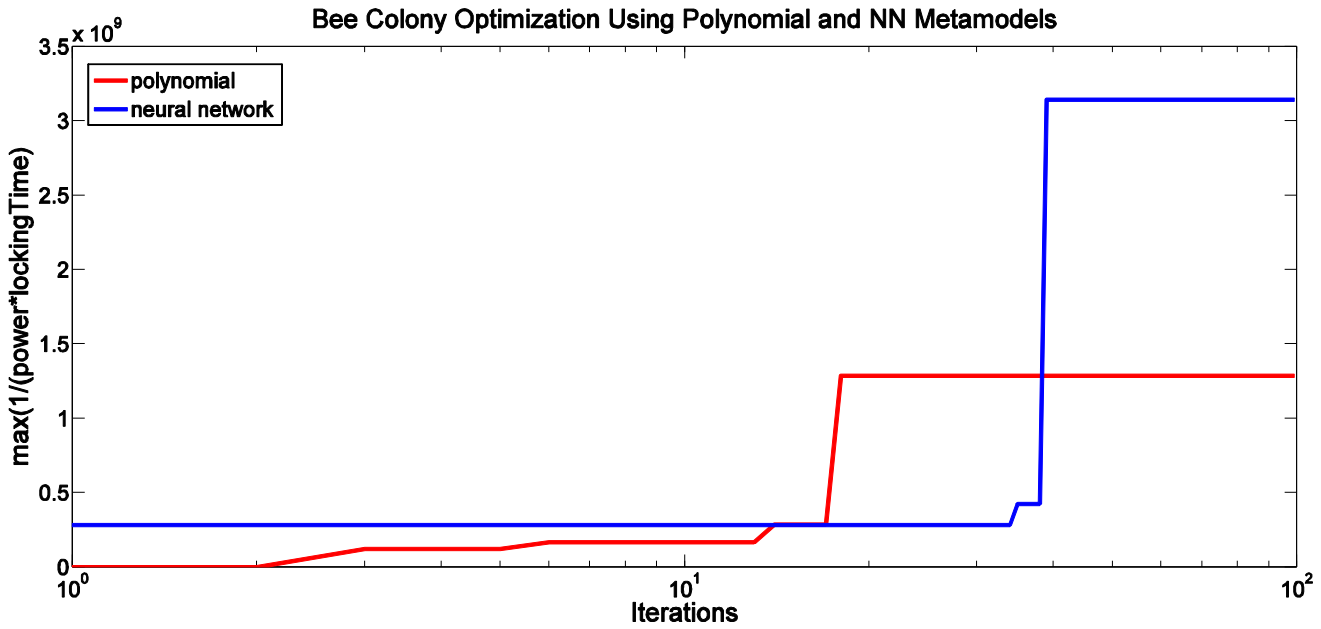
Algorithm 1 Proposed Bee Colony Optimization Algorithm.

Initialize maximum iterations $\leftarrow max_i$.
Set the boundaries for each parameter $P(i) \leftarrow [min, max]$.
$NumberBees \leftarrow$ **Define** the number of bees.
$buffer \leftarrow$ Number of close worker bees dispersal.
Initialize a matrix as follow: $bee_{matrix}(3, NumberBees) \leftarrow [workers, onlookers, scouts]$.
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 probability that the food source is good
        if (probability 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
    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.

Bee Colony Optimization

- Output frequency of PLL is used as a constraint function.

$$FoM = \left(\frac{1}{power \times lockingTime} \right)$$



Optimization Results

FoM	Baseline Power	Power optimality
Average Power	3.9 mW	3.9 mW
Locking Time	8.476 μs	3.3147 μs
Frequency	2.6909 GHz	2.7026 GHz

Conclusions

- A design flow for metamodeling using the neural networks is proposed
- A 180nm PLL was subjected to the proposed design flow
- Feed-forward NNs show better accuracy than polynomial models
- Bee colony optimization algorithm is implemented for circuit transistor sizing optimization.
- It is observed that there is no amount of hidden neurons that provide best result.
- 56% increase in accuracy is observed using NN over polynomial metamodels.
- On average 3.2% error is observed using NN.


Thank you !!!