Graph of Circuits with GNN for Exploring the Optimal Design Space (Supplementary)

Anonymous Author(s)
Affiliation
Address
email

In this document, we present a detailed report on two test cases examined in the paper: (i) Miller compensated Two-stage OTA and (ii) Three-stage Ring Oscillator. For each case, we provide a circuit schematic that showcases the different components named according to the input design parameters mentioned in the subsequent tables. The chosen Figure of Merit (FOM) formulas were carefully designed to strike a balance between multiple performance metrics, taking into consideration the specific requirements of each case. Additionally, a comprehensive time-complexity analysis is presented, supported by experimental evidence that clearly demonstrates the superior speed and efficiency of our proposed algorithms compared to other competitive methods.

1 Two-stage OTA with Miller compensation

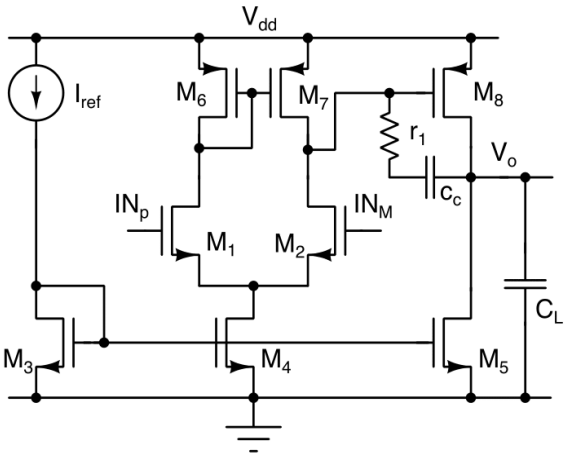

Figure 1: Schematic of Two-stage OTA with Miller Compensation

1.1 Theory

The operational transconductance amplifier (OTA) is a fundamental block in analog integrated circuits, which in selected example is comprised of two stages: a 5 transistor based differential input and single ended output differential amplifier as its first stage which is followed by a common-source amplifier as its second stage (output stage). It uses Miller compensation to ensure stability while simultaneously providing a high output swing. Consequently, the OTA becomes a versatile and indispensable building block in circuit designs such as operational amplifiers, analog filters, analog-to-digital converters (ADCs), voltage-controlled oscillators (VCOs), and data communication circuits.

The figure of merit (FOM) serves as a performance metric, providing an overall assessment of the OTA design's quality. Depending on the specific application and design objectives, the FOM for a two-stage Miller-compensated OTA can be defined in various ways. In this paper, we have formulated the FOM to encompass high gain, unity-gain bandwidth (UGB), phase margin(PM), and low noise

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

and power consumption. Each of these specifications holds equal weightage within the FOM formula, reflecting the importance of balancing and optimizing these key performance aspects.

$$\text{FOM} := \frac{\text{Power} \times \text{Noise}}{\text{Gain} \times UGB \times PM}$$

1.2 Tables

Following the designer's recommendations, we formulate a single-objective constrained optimization problem. To solve this problem, we utilize our proposed algorithms, namely EASCO and ASTROG, and evaluate their performance against other competing algorithms [3],[1].

$$\begin{aligned} \text{minimize} \quad & \textbf{FOM} \\ \text{s.t.} \quad & \text{Gain} \geq 50\,\text{dB} \\ & \text{Unit Gain Bandwidth} \geq 100\,\text{MHz} \\ & \text{Gain Margin} \geq 15\,\text{dB} \\ & \text{Phase Margin} \geq 45\,\text{deg} \\ & \text{Noise} \leq 600\,\mu\text{V}_{\text{rms}} \\ & \text{Power} \leq 900\,\mu\text{W} \end{aligned}$$

Miller Compensated Two-stage OTA (Optimization)							
Model	Gain (dB)	UGB (MHz)	GM (dB)	PM (deg)	Noise (nV)	Power(μW)	FOM
DE [3]	58.6	190	18.35	45.02	478.7	766.3	0.040
BO-EI[1]	**68.3**	**192**	18	**52.8**	**467**	**505.3**	**0.019**
EASCO	67.3	**205**	**18.53**	46.05	**465.5**	548.7	0.022
ASTROG	**80**	112	**24.84**	**52.48**	497.3	**495.5**	**0.021**

Table 1: comparison table for best case performance metrics with corresponding FOM

Through our observations, we note that the figure of merit (FOM) values obtained from each algorithm exhibit minimal differences. Notably, the BO-EI algorithm [1] yields the lowest FOM value, indicating superior performance. Following closely, the ASTROG algorithm demonstrates commendable results, placing second in terms of FOM.

We present the optimal input design parameters corresponding to the best FOM values achieved by each algorithm in the tables 2,3,4,5. Once the optimal design parameters are determined, we proceed to evaluate the corresponding performance metrics. To ensure their accuracy, we verify these metrics by comparing them with values obtained from simulations conducted using **Cadence**. This validation process enhances our confidence in the reliability of the obtained results.

Note: The labels in the below Optimized Input design parameters corresponds to the respective circuit labels shown in the figures. W_x:width of transistor X and L_x: Length of transistor X.

Design Parameters (ASTROG Algorithm)			
Parameter	Value	Parameter	Value
$Iref(\mu\text{A})$	20	$C_c(pF)$	0.3
$Vdd\,(V)$	2	$C_L(pF)$	2
$W_{M_1}(\mu\text{m})$	4	$r_1(k\Omega)$	4
$W_{M_2}(\mu\text{m})$	4	$W_{M_6}(\mu\text{m})$	7
$W_{M_3}(\mu\text{m})$	4	$W_{M_7}(\mu\text{m})$	7
$W_{M_4}(\mu\text{m})$	6	$W_{M_8}(\mu\text{m})$	90
W_{M_5}(nm)	35	$L_{M_{1-8}}$(nm)	500

Table 2: Optimized Input Design parameters with **ASTROG** algorithm.

Design Parameters (EASCO Algorithm)			
Parameter	Value	Parameter	Value
$Iref(\mu\text{A})$	30	$C_c(pF)$	0.3
$Vdd\,(V)$	1.3	$C_L(pF)$	2
$W_{M_1}(\mu\text{m})$	5	$r_1(k\Omega)$	4
$W_{M_2}(\mu\text{m})$	5	$W_{M_6}(\mu\text{m})$	11
$W_{M_3}(\mu\text{m})$	3.65	$W_{M_7}(\mu\text{m})$	11
$W_{M_4}(\mu\text{m})$	10	$W_{M_8}(\mu\text{m})$	90
W_{M_5}(nm)	40	$L_{M_{1-8}}$(nm)	500

Table 3: Optimized Input Design parameters with **EASCO** algorithm.

To conduct our analysis, we employed labeled datasets consisting of 100 samples for each algorithm. In order to alleviate computational strain, we limited the number of runs for each algorithm to 100.

Design Parameters (BO-EI Algorithm)			
Parameter	Value	Parameter	Value
$Iref(\mu\text{A})$	30	$C_c(pF)$	0.32
$Vdd\,(V)$	1.3	$C_L(pF)$	2
$W_{M_1}(\mu\text{m})$	5	$r_1(k\Omega)$	4
$W_{M_2}(\mu\text{m})$	5	$W_{M_6}(\mu\text{m})$	11
$W_{M_3}(\mu\text{m})$	4	$W_{M_7}(\mu\text{m})$	11
$W_{M_4}(\mu\text{m})$	10	$W_{M_8}(\mu\text{m})$	90
$W_{M_5}(\text{nm})$	40	$L_{M_{1-8}}(\text{nm})$	500

Table 4: Optimized Input Design parameters with **BO-EI**[1] algorithm.

Design Parameters (DE Algorithm)			
Parameter	Value	Parameter	Value
$Iref(\mu\text{A})$	30	$C_c(pF)$	0.3
$Vdd\,(V)$	2	$C_L(pF)$	2
$W_{M_1}(\mu\text{m})$	5	$r_1(k\Omega)$	4
$W_{M_2}(\mu\text{m})$	5	$W_{M_6}(\mu\text{m})$	11
$W_{M_3}(\mu\text{m})$	3.69	$W_{M_7}(\mu\text{m})$	11
$W_{M_4}(\mu\text{m})$	9.33	$W_{M_8}(\mu\text{m})$	81.5
$W_{M_5}(\text{nm})$	40	$L_{M_{1-8}}(\text{nm})$	500

Table 5: Optimized Input Design parameters with **DE**[2] algorithm.

To expedite the optimization process and reduce computational overhead, we utilized pre-trained surrogate models. These models effectively minimized the reliance on resource-intensive **SPICE** calls, enhancing the efficiency of our algorithms.

Detailed timing comparisons for each algorithm are presented in a subsequent section, providing valuable insights into the computational aspects of our approach.

2 Three-stage Ring Oscillator

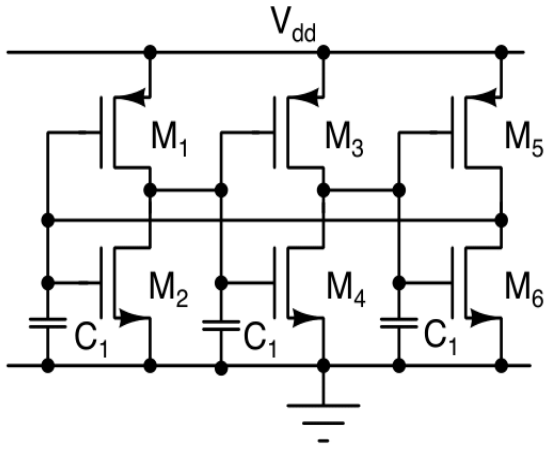

Figure 2: Schematic of Three-stage Ring Oscillator

2.1 Theory

A ring oscillator is a self-sustaining oscillator that generates a continuous oscillating output signal without the need for an external input. It operates by utilizing an odd number of delay stages, typically implemented using inverters, arranged in a closed loop or ring configuration. These stages operate in the nonlinear region, contributing to the oscillator's functionality.

Each delay stage introduces a propagation delay, which establishes a positive feedback loop, leading to sustained oscillations. As the signal propagates through each stage, it undergoes a phase shift due to the inherent propagation delay.

The collective delay introduced by each stage, combined with the propagation delay through the inverters, determines the frequency of oscillation produced by the ring oscillator. Additionally, the amplitude of the oscillation is influenced by factors such as the supply voltage and the characteristics of the inverters employed in the circuit.

The choice of a specific Figure of Merit (FOM) for a ring oscillator depends on the priorities and design objectives of the application at hand. For low-power applications, minimizing power consumption (P_{dis}) may be the primary concern. In contrast, for high-performance applications, increasing oscillation frequency (f_{osc}), reducing Jitter ($J(t)$), or optimizing the speed/power trade-off

may take precedence. The selection of the FOM should align with the specific requirements and goals of the design.

The Figure of Merit (FOM) for a ring oscillator can be calculated using the following formula, where fm represents the offset frequency set to 1MHz. We referred [2, 4] while choosing an appropriate FOM formula.

$$\text{FOM} := J(t) - 20\log\left(\frac{f_{\text{osc}}}{f_m}\right) + 10\log\left(\frac{P_{\text{dis}}}{1\,\text{mW}}\right)$$

2.2 Tables

Following the designer's recommendations, we formulate a single-objective constrained optimization problem. To solve this problem, we utilize our proposed algorithms, namely EASCO and ASTROG, and evaluate their performance against other competing algorithms [3],[1].

$$\begin{aligned} \text{minimize} \quad & \textbf{FOM} \\ \text{s.t.} \quad & \text{Frequency} \geq 1000\,\text{MHz} \\ & \text{RMS Jitter} \leq 10\,\text{pS} \\ & \text{Delay} \leq 200\,\text{pS} \\ & \text{Power} \leq 600\,\mu\text{W} \end{aligned}$$

Ring Oscillator - Three stage (Optimization)					
Model	Frequency (MHz)	rms Jitter (pS)	Delay (pS)	Power (μW)	FOM
DE [3]	1265	0.99	131.7	**256**	-66.96
BO-EI[1]	1217	**7e-7**	136.9	**66.4**	**-73.48**
EASCO	**1444**	**2.2e-7**	**115.4**	335.2	**-67.94**
ASTROG	**1740**	1.2	**95.7**	502.3	-67.27

Table 6: comparison table for best case performance metrics with corresponding FOM

The table above (Table 6) presents the optimal Figure of Merit (FOM) values obtained for each algorithm, along with the corresponding performance metric values. Among the algorithms, BO-EI[1] achieved the highest FOM value. However, when considering the overall performance metrics, EASCO and ASTROG outperformed the other algorithms. These algorithms demonstrated better performance across various metrics, indicating their superiority in terms of overall performance.

Ring Oscillator - Three stage (Optimized Parameters)						
Model	Vdd (V)	c1 (fF)	W1 (μm)	W2 (μm)	L1 (nm)	L2(nm)
DE [3]	1	3	7.72	12	300	200
BO-EI[1]	1	3	2.4	2	300	200
EASCO	1	3	12	9.48	300	200
ASTROG	1.4	3	4.5	4.5	300	400

Table 7: Optimized Input Design parameters with **DE**[3], **BO-EI**[1], **EASCO** and **ASTROG**

Table 7 provides the optimal input design parameters corresponding to the best-case Figure of Merit (FOM) for each scenario. These parameters represent the input configurations that yielded the highest FOM values for each case.

3 Time Complexity for each algorithm

Let's deconstruct each algorithm and thoroughly analyze each step to enhance our comprehension of the time complexity linked to each of them.

3.1 Differential Evolution (DE) algorithm

The overall time complexity of the algorithm can be expressed as:

$$O(n + E * P * D * iter + S + R)$$

where, n: The number of initial samples, E: The number of epochs, P: The population size in the differential evolution algorithm, D: The number of dimensions in the problem, iter: The number of iterations in the differential evolution algorithm, S: The number of samples to predict using surrogate models, R: The number of rows in the X and y matrices.

This time complexity analysis takes into account the various steps involved in the algorithm, such as initializing the dataset and variables, iterating through epochs, running the differential evolution algorithm, predicting new values using surrogate models, and updating the X and y matrices. Please note that this is an approximation, and the actual time complexity may vary depending on specific implementation details and the complexity of the surrogate models used.

3.2 Bayesian Optimization-Expected Improvement (BO-EI) algorithm

The time complexity of using a Gaussian regressor in Bayesian optimization can be summarized as follows:

1. Initializing the dataset and variables: $O(n)$

2. Iterating through epochs: $O(epochs)$

3. Gaussian Process (GP) Regression:

3.1 Computing the kernel matrix: $O(N^3)$

3.2 Inverting the kernel matrix: $O(N^3)$

4. Predicting new values: $O(N^2)$

5. Predicting new values using surrogate models: $O(S)$; S is the number of samples on which prediction will be performed.

6. Updating the X and y matrices: $O(R)$; R is number of rows in X and y.

Overall, the time complexity can be approximated as:

$$O(n + epochs * O(N^3) + S + R)$$

3.3 Efficient Analog Sizing via Constrained Optimization (EASCO) algorithm

The given code incorporates pre-trained surrogate models, so the training time of the models is not utilized in the code.

1. **Initializing the dataset and variables**: $O(nodes)$

2. **Graph Learning**:

2.1 Creating sparse matrices: $O(nodes^2)$,

2.2 Training loop over epochs:

2.2.1 Computing gradients: $O(nodes^2)$

2.2.2 Updating weight vector: $O(nodes^2)$

3. **Label prediction using GNNs** : $O(nodes)$

4. **Differential Evolution**:

4.1 Generating new solutions: $O(popsize * bounds)$

4.2 Evaluating the objective function: $O(popsize)$

Thus, the overall time complexity can be summarised as follows:

$$O(nodes^2 + popsize * bounds)$$

3.4 Analog Sizing through Real-time Online Graphs (ASTROG) algorithm

The given code incorporates pre-trained surrogate models, so the training time of the models is not utilized in the code. The time complexity of the given algorithm can be analyzed as follows:

1. **Initialization**: Generating random weight matrix of size $nodes^2$: $O(nodes^2)$, Normalizing feature matrix: $O(nodes * features)$.

2. **Graph Learning**:

2.1 Creating sparse matrices: $O(nodes^2)$,

2.2 Training loop over epochs:

2.2.1 Computing gradients: $O(nodes^2)$

2.2.2 Updating weight vector: $O(nodes^2)$

3. **Rearranging data based on performance ranking**: $O(nodes * features)$

4. **Differential Evolution**:

4.1 Generating new solutions: $O(popsize * bounds)$

4.2 Evaluating the objective function: $O(popsize)$

4.3 Graph Learning with updated data:

4.3.1 Creating sparse matrices: $O(nodes^2)$

4.3.2 Training loop over epochs:

4.3.2.1 Computing gradients: $O(nodes^2)$

4.3.2.2 Updating weight vector: $O(nodes^2)$

4.3.2.3 Predicting values using neural networks: Forward pass through the models: $O(nodes)$.

Overall, the time complexity of the algorithm can be approximated as:

$$O(nodes^2 + epochs * nodes^2 + popsize * bounds + nodes * features)$$

The below graphs display real-time estimations of the computational time for each algorithm. These graphs serve as experimental evidence, showcasing the relative speed of our algorithms. The bar plots illustrate the performance of four algorithms, namely DE, BO-EI, EASCO, and ASTROG, for two different test cases.

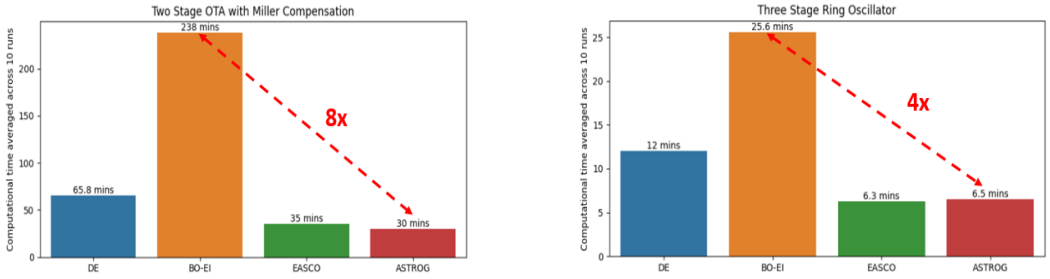

Figure 3: The bar plots in Figure (i) compare computational times for different algorithms in the **Miller-compensated Two-stage OTA**. Figure (ii) does the same for the **Three-stage Ring Oscillator**. The bar heights represent the computational times, and the time values are displayed on top of each bar for precise comparison. These plots provide real-time estimates and experimental evidence of algorithm speeds.

References

[1] Shady A. Abdelaal, Ahmed Hussein, and Hassan Mostafa. A bayesian optimization framework for analog circuits optimization. In *2020 15th International Conference on Computer Engineering and Systems (ICCES)*, pages 1–4, 2020.

[2] Teruki Matsuba, Nobukazu Takai, Masafumi Fukuda, and Yusuke Kubo. Inference of suitable for required specification analog circuit topology using deep learning. In *2018 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pages 131–134, 2018.

[3] Samrat L. Sabat, K.Shravan Kumar, and Siba K. Udgata. Differential evolution and swarm intelligence techniques for analog circuit synthesis. In *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pages 469–474, 2009.

[4] Khalil Yousef. A low phase noise, high figure of merit, 3.1 ghz–3.5 ghz ring oscillator using edge injection technique. *2017 Japan-Africa Conference on Electronics, Communications and Computers (JAC-ECC)*, pages 37–40, 2017.