Dean&Francis

ISSN 2959-6157

Parasitic Parameter Sensitivity Analysis and Simulation Evaluation Based on Cadence SKILL

Yetong Li

Abstract:

Parasitic effects caused by interconnects have become increasingly critical in modern analog integrated circuits, especially as technology nodes in the semiconductor industry continue to advance. This paper presents an automated sensitivity analysis and simulation evaluation framework that addresses these challenges using Cadence SKILL and the Virtuoso platform. The purpose of this research is to identify and quantify the impact of interconnect parasitic parameters on circuit performance, enabling more efficient post-layout optimization. The proposed method incorporates automated parasitic insertion, simulation management, and statistical evaluation. Morris-based sensitivity analysis is used to determine the significance of individual parasitic elements, while a two-dimensional interaction analysis based on regression identifies pairs of critical variables with compensatory behavior. The experimental test case is a bandgap reference circuit implemented in TSMC 65nm technology, and key performance indicators such as gain, bandwidth, gain-bandwidth product, phase margin, and power supply rejection ratio are evaluated to comprehensively score the sensitivity indicators under various performance conditions. The results demonstrate that the proposed method can effectively identify critical parasitic parameters affecting the performance and maintain high simulation accuracy while significantly reducing complexity by ignoring low-sensitivity parameters. This research offers an efficient and scalable analysis approach for post-layout simulation modeling of analog circuits, showing strong potential for practical engineering applications.

Keywords: Parasitic Parameters, SKILL Language, Sensitivity Analysis, Morris Method, Interaction Effects

ISSN 2959-6157

1 Introduction

As integrated circuits continue to evolve, their miniaturization and increased complexity bring challenges in accurately modeling. In the early days of integrated circuit technology, the speed of circuits was primarily determined by the switching speed of individual transistors, and the parasitic effects of interconnections could be largely negligible. As IC fabrication has advanced into increasingly smaller technology nodes, the continuous scaling down of feature sizes and the growing density of devices have significantly increased the density and structural complexity of interconnects. While this trend has improved integration and performance, it has also made the parasitic resistance, capacitance, and inductance effects generated by interconnects increasingly prominent. These parasitic effects now exert a substantial impact on critical performance metrics such as timing, power consumption, noise, and stability, and have become one of the major factors limiting circuit performance[1]. Parasitic effects are critical considerations in the post-layout simulation phase. However, when faced with the large number of parasitic parameters introduced by complex interconnections, indiscriminately introducing all parasitic variables leads to a significant increase in simulation complexity and runtime. Therefore, identifying the key parasitic parameters that most significantly impact circuit performance and reducing the dimensionality of simulation inputs without compromising accuracy, is of considerable research importance in contemporary analog circuit design and verification processes.

Sensitivity analysis, as a fundamental approach for evaluating the influence of model input parameters on output responses, has been widely adopted in the field of EDA. Common sensitivity analysis methods used in engineering and scientific computing include local sensitivity analysis, variance-based sensitivity analysis, and the Morris method. Among them, the Morris method[2] is particularly favored for its high computational efficiency and simplicity, making it well-suited for preliminary screening in high-dimensional models. In complex systems, interactions among parameters can significantly influence overall system performance. Traditional sensitivity analysis methods often fail to account for such interaction effects, leading to an incomplete or misleading understanding of system behavior. To address this limitation, researchers have proposed various approaches to analyze and quantify parameter interactions. For instance, the Sobol method[3] assesses higher-order sensitivity indices to evaluate the contribution of parameter interactions. Regression analysis is another commonly employed method; by incorporating interaction terms into regression models, it becomes possible to quantitatively evaluate how the interplay between parameters affects the system output.

This study aims to address the following issues in the circuit design simulation phase: how to algorithmically and systematically evaluate the impact of a large number of parasitic parameters; how to identify high-sensitivity variables and eliminate redundant parameters; and, on this basis, how to further explore the interactions among variables to support symmetry-aware design and optimize matching strategies. In summary, we proposed and implemented an automated framework for parasitic parameter simulation and sensitivity evaluation based on the Cadence Virtuoso and SKILL language:

• A set of SKILL scripts was developed to automate the insertion of parasitic variables, enabling procedural insertion of parasitic parameters of metal interconnect, parameter configuration, and schematic updating.

• Combining SKILL-based simulation control scripts with the Morris method, we performed quantitative sensitivity analysis to assess the comprehensive impact of parasitic parameters on circuit performance. Key variables with the most significant influence on performance metrics were identified, and comparative simulations were conducted before and after processing to validate the effectiveness of parasitic compression.

• Combining two-dimensional parameter scanning with regression-based interaction effect analysis, a method using the SKILL language to analyze interaction effects between pairs of parasitic parameters.

2 Automated Insertion of Parasitic Variables

To eliminate the tedious and error-prone process of manually inserting parasitic devices into each node, we developed a comprehensive automation script based on SKILL language[4]. The script enables batch insertion of parasitic resistors and capacitors into the schematic while ensuring correct connectivity. To achieve automated modeling of parasitic parameters, strict insertion rules must be defined and implemented through SKILL-based algorithmic logic. According to the modeling requirements, a series resistor and a parallel capacitor to ground are inserted between each network node and its connected device terminals, to emulate the series resistance in metal interconnects and the parasitic capacitance from metal wires to ground.

The script first retrieves all net objects from the extracted schematic of the target cellView. It iterates over all nets, and for each net, it extracts all connected device terminals. For each terminal, a set of resistors and capacitors is inserted, along with the creation of a new intermediate net node to implement the required series and parallel connections. To avoid naming conflicts and facilitate parameter sweeps in later simulations, the parameters for resistor and capacitor instances are dynamically generated using a counter (e.g., “r1”, “c1”). During resistor instance creation, one terminal is connected to the original net, and the other terminal is connected to a newly created net. Similarly, for capacitor instances, one terminal is connected to ground, while the other is connected to the same new net node used for the resistor. All newly created instances are placed in sequence above the original circuit layout. For clarity, all existing wire shapes in the cellView are first deleted. After creating instances and their terminals, wire stubs are generated for all devices, and labels are added to the wires based on the net attributes of the instance terminals, facilitating visual inspection and debugging.

Algorithm 1: Automated Insertion of Parasitic Variables

```
   Input: Target schematic cellView ($cv$)
1  Delete all wire shapes
2  foreach $net \in cv \to nets$ do
3      if $number\ of\ connected\ instance\ terminals\ net \to instTermCount = 2$ then
4          Create parasitic device instance: $dbCreateParamInst()$
5          Create new net: $n\_net = dbCreateUniqueNamedNet()$
6          Modify net property of one instance terminal: $insTerm \to net = n\_net$
7          Create R/C instance terminals and set net properties: $dbCreateInstTerm()$
8      else
9          foreach $instance\ terminal\ insTerm \in net \to allInstTerms$ do
10             Create parasitic device instance: $dbCreateParamInst()$
11             Create new net: $n\_net = dbCreateUniqueNamedNet()$
12             Modify instance terminal's net property: $insTerm \to net = n\_net$
13             Create R/C instance terminals and set net properties: $dbCreateInstTerm()$
14         end
15     end
16 end
17 Select all instances in $cv$: $schHiSelectAll()$
18 Generate wire stubs for all instances: $schHiCreateWireStubs()$
19 foreach $instance\ inst \in cv \to instances$ do
20     foreach $instance\ terminal\ instTerm \in inst \to instTerms$ do
21         foreach $terminal \in inst \to master \to terminals$ do
22             foreach $pin \in terminal \to pins$ do
23                 if $pin \to net \to name = instTerm \to name$ then
24                     Transform pin's bounding box to schematic coordinates:
                        $pinBox = dbTransformBBox()$
25                     Find overlapping wire stub: $wire = dbShapeQuery()$
26                     foreach $child\ object\ child \in wire \to children$ do
27                         if $child \to objType =$ “label” then
28                             Update label text: $child \to theLabel = instTerm \to net \to name$
29                         end
30                     end
31                 end
32             end
33         end
34     end
35 end
```

3 Automated simulation and parameter sensitivity analysis

The test object used in this study is a bandgap voltage reference designed using the TSMC 65nm CMOS process , with an operating voltage of 1.8V and an operating temperature range of -40°C to 125°C. Simulation setup and control are carried out using Ocean Script. OCEAN, a subset of the SKILL language, enables automated configuration of the simulation environment within Cadence,

ISSN 2959-6157

allowing simulations to be executed from the command line with full control from start to finish. This makes it particularly suitable for large-scale parameter sweeps and Monte Carlo analyses.

With Ocean Script, users can selectively save only the simulation data of interest and directly store the results in individual or batch files, which not only conserves storage space but also facilitates post-simulation review and further data analysis. To reduce manual intervention and streamline the evaluation process, we write Ocean Script to configure simulations for various performance metrics, define parameter sweeping strategies, manage data storage, and perform preprocessing. The script is integrated with SKILL-based sensitivity analysis algorithms, enabling the construction of a unified SKILL-driven framework for automated simulation and evaluation.

Since SKILL is a proprietary language of the Cadence Virtuoso and lacks the advanced data structures and randomization libraries found in languages like Python, we need to manually implement the sampling logic, simulation control, and analytical algorithms. Specifically, we used loop control statements and the append() function to construct a list of parasitic parameters and iterate through all variables. Considering that parasitic parameter names follow a structured naming convention of "parameter type + index" , we classified parameter types by inspecting the first character of their names and assigned corresponding value ranges accordingly. These value ranges were determined based on the specifications of the employed process design kit.

We use the Morris method[2] to analyze the sensitivity of parasitic parameters. The Morris method is an improved global sensitivity analysis technique derived from the local one-at-a-time method (OAT). It is particularly well-suited for scenarios involving a large number of input variables and limited computational resources, as it enables efficient identification of key parameters. The core idea of the Morris method is to compute the elementary effect (EE) of each parameter by sequentially varying one input at a time and observing its impact on the model output, then statistically analyzing the mean (μ) and standard deviation (σ) to assess the variable's impact on the model output.For multi-parameter models, each input variable is first normalized to the unit interval [0,1] and discretized into an evenly spaced set of values with a resolution level defined by the discretization level p. A trajectory is then constructed within the parameter space Ω , along which the EE values are computed[5]. The number of simulations required to complete one round of EE value calculations along the trajectory path is p, while the number of simulations required for random sampling is $2(p-1)$. The method of sampling along the trajectory path requires fewer simulations.

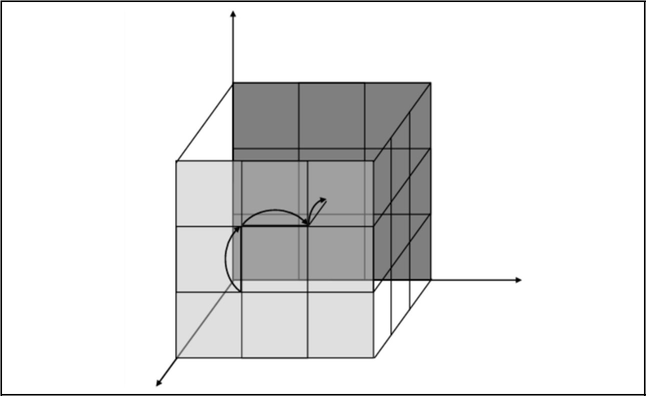
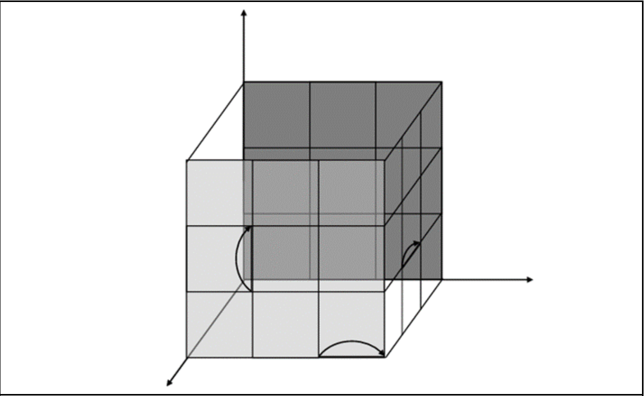
	
(a) Sampling points selected randomly; six simulations are required to obtain the EE values of the three variables.	(b) Sampling points selected along the trajectory path; only four simulations are required to obtain the EE values of the three variables.

Figure 1: Parameter space of a three-variable model, $p = 4$

Trajectory paths are used to represent disturbances in variables, thereby calculating the EE value for each input variable. The elementary effect is defined by the following equation:

$$EE_i = \frac{y(x_1, x_2, \ldots, x_i + \Delta, \ldots, x_k) - y(\mathbf{x})}{\Delta} \tag{1}$$

In this case, Δ is the step size for each variable perturbation.To define the path of each trajectory, Morris used a series of matrices to construct the final trajectory matrix B^* :

$$B^* = (\mathbf{J}_{m,1}\mathbf{x}^* + \frac{\Delta}{2}\left[(2B - \mathbf{J}_{m,k})D^* + \mathbf{J}_{m,k}\right])P^* \tag{2}$$

Among them, B is a $(k+1) \times k$ sampling matrix, with each column having two rows that differ only in the i-th item; D^* is a diagonal matrix, whose diagonal elements take values of +1 or -1 with equal probability, allowing the variable to increase or decrease by Δ; P is a $k \times k$ permutation matrix, with each row and column containing only one 1; $\mathbf{x}^*$ is a reference point vector randomly selected from $[0,1-\Delta]$; $\mathbf{J}_{m,k}$ is an m×k identity matrix with all entries equal to 1.Each column (variable) in B* undergoes a perturbation of $+\Delta$ or $-\Delta$ in a specific row. Multiple trajectory matrices B^* can be combined to form the final experimental matrix X.

Scale the columns of X to the range of variation of the input variables, calculate the EE value of each variable on r paths, and calculate the mean μ_i and standard deviation σ_i of the basic effect of the input variables. To eliminate the influence of monotonic effects, mean the absolute value μ_i^* can be used to evaluate the sensitivity of variables:

$$\mu_i^* = \frac{1}{r}\sum_{j=1}^{r} ?\left|EE_i^{(j)}\right| \tag{3}$$

Since we need to comprehensively score the sensitivity indicators under various performance conditions, it is necessary to normalize the performance indicators to make the sensitivity indicators balanced and comparable. We modify the calculation formula for the basic effect in the Morris method, taking the performance indicators obtained from the simulation with all parameters set to the default value as the baseline value. The modified calculation formula for the basic effect is as follows:

$$EE_i = \frac{y(x_1, x_2, \ldots, x_i + \Delta, \ldots, x_k) - y(\mathbf{x})}{\Delta \cdot y(\mathbf{x}^{(0)})} \tag{4}$$

The SKILL program was written based on the basic principles of the Morris method:

Algorithm 2: Matrix inversion

```
   Input: A list of 4 sublists, each containing 4 elements, denoted as matrix $X$
1  Augment each sublist to construct an augmented matrix
2  for $i = 1$ to 4 do
3      if the pivot element is zero: $nth(i\ nth(i\ X) = 0$ then
4          for $j = i + 1$ to 4 do
5              if $nth(j\ nth(i\ X) = 0 \neq 0$ then
6                  Swap the $i$-th and $j$-th rows
7              end
8          end
9      end
10     Normalize the $i$-th row by $nth(i\ nth(i\ X)$:
       $nth(k\ nth(i\ X) \leftarrow nth(i\ nth(k\ X)/nth(i\ nth(i\ X)$
11     for $j = 1$ to 4 do
12         if $j \neq i$ then
13             $factor \leftarrow nth(i\ nth(j\ X)$ for $k = 1$ to 8 do
14                 $nth(k\ nth(j\ X) \leftarrow nth(j\ nth(k\ X) - factor \cdot nth(k\ nth(i\ X)$
15             end
16         end
17     end
18 end
19 return The right 4 columns of the augmented matrix as $X^{-1}$
```

After running the script, it will output a sorted table including parameter names, sensitivity metrics for each parameter, and a comprehensive sensitivity score obtained by summing the sensitivity metrics.

4 Two-dimensional interaction effect analysis based on regression models

Interaction refers to the influence on the output caused by the combined behavior of two or more variables. In a multi-parameter model, input variables may not only have direct effects on the output, but can also interact in a synergistic or antagonistic manner, resulting in significant interaction effects that influence system performance. A common approach to analyzing interaction effects between continuous variables is to fit a regression model that includes interaction terms. For two continuous independent variables x_1 and x_2, their interaction can be quantitative-

ISSN 2959-6157

ly analyzed using a regression model that includes both the main effects and their interaction term x_1x_2, typically formulated as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \grave{o} \qquad (5)$$

where β_1 and β_2 represent the main effect coefficients of x_1 and x_2, respectively, while β_3 represents the interaction effect coefficient. The absolute value of β_3 indicates the strength of the interaction between x_1 and x_2; the larger the value, the more significant the interaction effect. Assuming that x_1 and x_2 influence the output in the same direction—that is, β_1 and β_2 share the same sign—a positive β_3 suggests an antagonistic effect. This means the joint variation of x_1 and x_2 may attenuate their individual effects on the output. In circuit performance simulations, this typically manifests as a smaller performance degradation when both x_1 and x_2 increase or decrease simultaneously, compared to changing only one of them. On the other hand, if x_1 and x_2 have opposite effects on the output, it is possible to devise a coordinated variation scheme based on the fitted coefficients. Suppose the parameter pair (x_1, x_2) is varied such that $x_2 = kx_1$; then, according to the regression regression, when $k = -\beta_1 / \beta_2$, the linear main effects cancel each other out. the output is dominated by the quadratic term, and for small parameter variations, this approach can effectively suppress output fluctuations.

The parameters of the regression model are typically estimated using the least squares method. The objective of the least squares method is to determine a set of parameters $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3$ that minimize the sum of squared residuals between the predicted and actual values.

Assuming a sample size of n, the design matrix X is constructed as follows:

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{11}x_{12} \\ 1 & x_{21} & x_{22} & x_{21}x_{22} \\ ? & ? & ? & ? \\ 1 & x_{n1} & x_{n2} & x_{n1}x_{n2} \end{bmatrix}$$

where the first column corresponds to the intercept term, the second and third columns to the linear terms of x_1 and x_2, respectively, and the fourth column to the interaction term. The regression model is then reformulated in matrix form as:

$$Y = X\beta + \grave{o} \qquad (6)$$

The residual sum of squares (RSS) is defined as:

$$RSS = (Y - X\beta)^T (Y - X\beta) \qquad (7)$$

To obtain the least squares estimate of β, we take the derivative of RSS with respect to β and set it to zero:

$$\frac{\partial RSS}{\partial \beta} = -2X^T (Y - X\beta) = 0 \qquad (8)$$

Solving this equation yields the least squares estimator:

$$\hat{\beta} = (X^T X)^{-1} X^T Y \qquad (9)$$

To investigate the interaction effects between variables, we need to perform a combinatorial sweep of all parasitic parameters. Based on the results of the sensitivity analysis in the previous chapter, we select only the the high-sensitivity parasitic parameters that significantly impact performance as the focus of this study. Consistent with the previous chapter, we use OceanScript to control the two-dimensional parameter sweep simulations. By iterating through the list of parameters (x_i, x_j) using a nested loop, simulation setup, execution, and result extraction can be automatically completed. Since the coefficient fitting in the linear regression model involves matrix operations, and because the the SKILL language is essentially a Lisp-based language that operates primarily through lists and lacks built-in matrix computation libraries, all matrix operations must be implemented manually.

1. Constructing the design matrix

Algorithm 3: Constructing the design matrix

```
  Input: List of sample points $paramspairs = \{(x_1^{(i)}, x_2^{(i)})\}_{i=1}^n$
  Output: Design matrix $X \in \mathbb{R}^{n\times 4}$
1 Initialize an empty list $X = list()$
2 foreach $(x_1, x_2) \in paramspairs$ do
3 |  Construct a row and append it to $X$:$X = append1(X\ (1\ x_1\ x_2\ x_1x_2))$
4 end
5 return $X$
```

As shown in Algorithm 1, the mapcar() function is used to apply an operation to each sample pair (x_1, x_2) : it extracts the first dimension x_1 and the second dimension x_2 and constructs a new list $(1 x_1 x_2 x_1 x_2)$. These rows are then aggregated to form a list-based design matrix for multiple linear regression with interaction terms.

2. Matrix transposition

The mpcar() function in SKILL can be used to sequentially operate on elements of each sublist within a list collection. By combining the mpcar() function with list generation techniques, it is possible to extract elements from sublists in a systematic manner, thereby achieving functionality analogous to matrix transposition.

3. Matrix multiplication

Given two matrices X_1 and X_2 to be multiplied, the first step is to transpose X_2. Then, by applying the mapcar() function in a nested manner, one can multiply corresponding elements of each sublist in X_1 and the transposed X_2, and compute their sum.

4. Matrix inversion

Since the inversion in Equation 9 only involves a 4×4 matrix, it is sufficient to implement the inversion for a 4×4 matrix. This can be accomplished using the Gauss-Jordan elimination method.

Algorithm 4: Matrix inversion

```
Input: A list of 4 sublists, each containing 4 elements, denoted as matrix $X$
Augment each sublist to construct an augmented matrix
for $i = 1$ to 4 do
    if the pivot element is zero: $nth(i\ nth(i\ X) = 0$ then
        for $j = i + 1$ to 4 do
            if $nth(j\ nth(i\ X) = 0 \neq 0$ then
                Swap the $i$-th and $j$-th rows
            end
        end
    end
    Normalize the $i$-th row by $nth(i\ nth(i\ X)$:
      $nth(k\ nth(i\ X) \leftarrow nth(i\ nth(k\ X)/nth(i\ nth(i\ X)$
    for $j = 1$ to 4 do
        if $j \neq i$ then
            $factor \leftarrow nth(i\ nth(j\ X)$ for $k = 1$ to 8 do
                $nth(k\ nth(j\ X) \leftarrow nth(j\ nth(k\ X) - factor \cdot nth(k\ nth(i\ X)$
            end
        end
    end
end
return The right 4 columns of the augmented matrix as $X^{-1}$
```

Once matrix operations are implemented, the regression coefficients can be solved directly according to Equation 9. By integrating the simulation loop with the analysis code, the regression models for all variable pairs can be computed.

5 Result

We conducted sensitivity analysis on both the core circuit and the operational amplifier (op-amp) of the bandgap reference.

1. Core circuit The cellView contains 95 pairs of parasitic parameters after insertion. We set the discretization level $p = 10$ and the number of trajectories $r = 10$, and analyzed its temperature coefficient. The sensitivity score threshold is set to 0.1. After filtering, there are 19 highly-sensitivity parameters in the core circuit.

2. Operational amplifier The cellView contains 47 pairs of parasitic parameters after insertion.We set the discretization level $p = 10$ and the number of trajectories $r = 10$, and analyzed its temperature coefficient. The sensitivity score threshold is set to 0.01. After filtering, there are 19 highly-sensitivity parameters and 75 low-sensitivity parameters in the op-amp.

ISSN 2959-6157

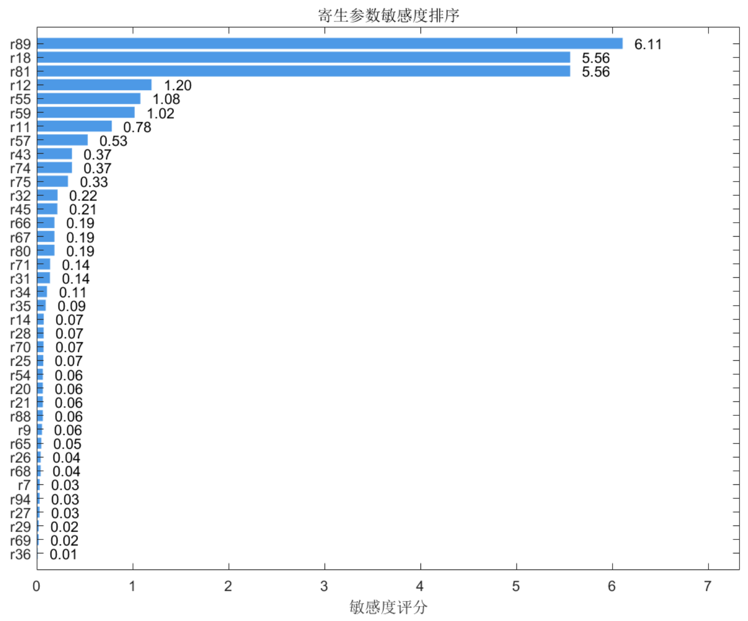


Figure 2: Sensitivity ranking of parasitic parameters of the core circuit

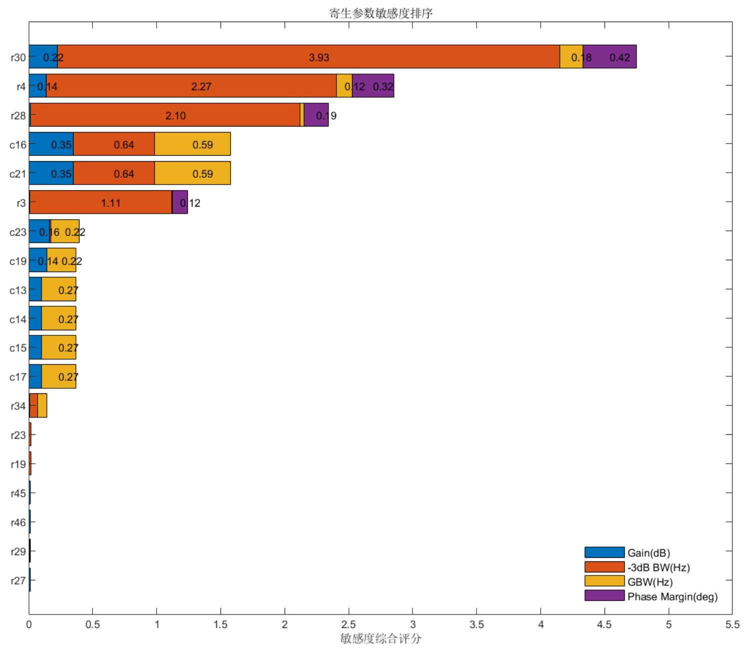
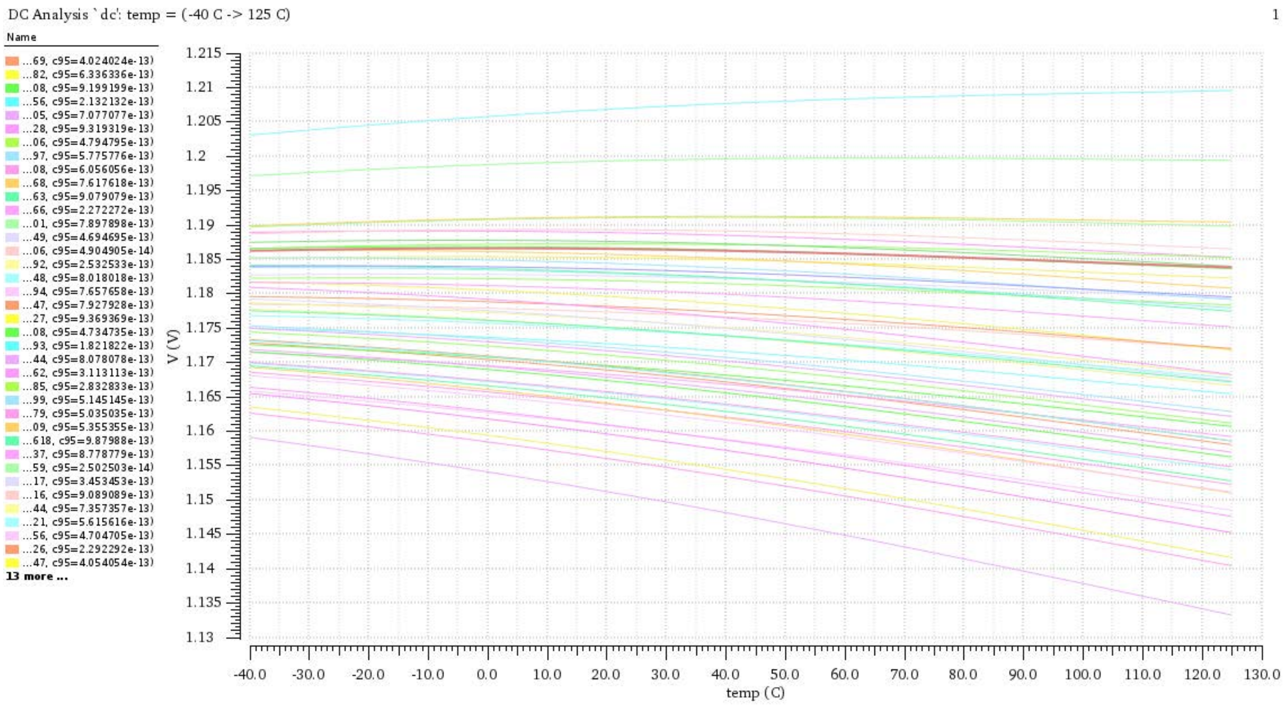


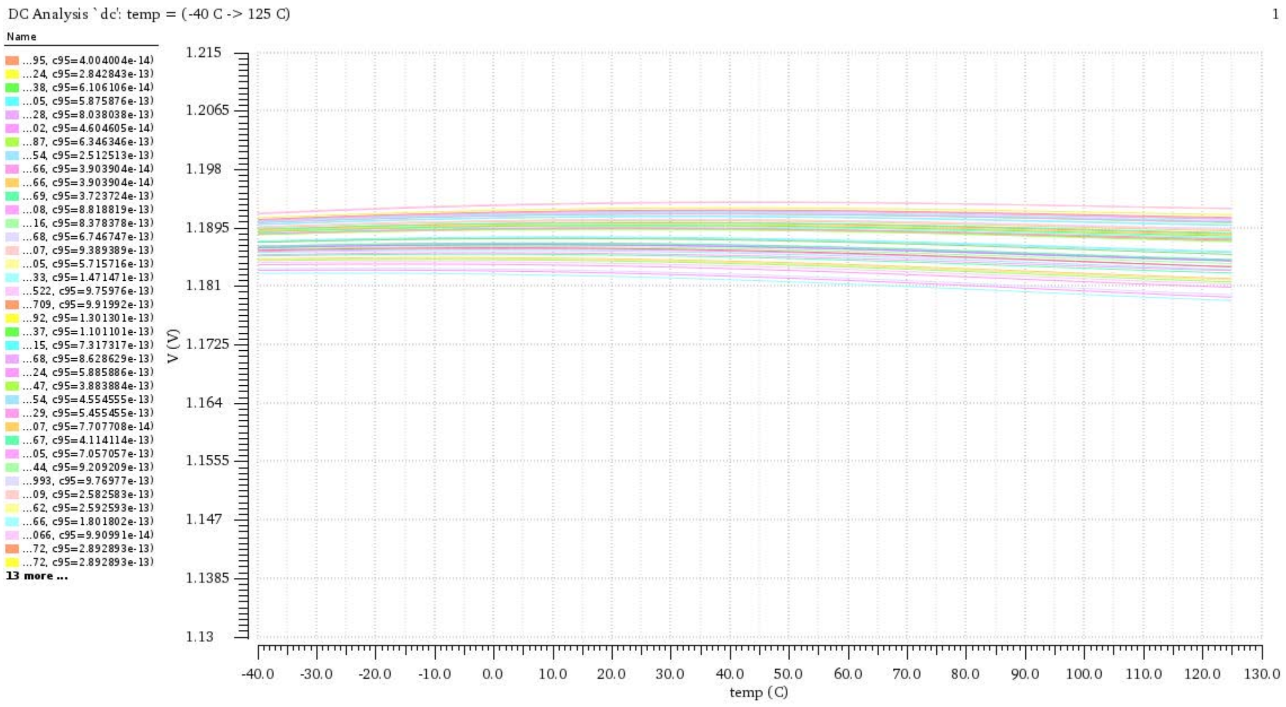
Figure 3: Sensitivity ranking of parasitic parameters of the op-amp

To verify the effectiveness and accuracy of the simulation framework and sensitivity analysis method, we performe multiple sets of Monte Carlo simulations on the test circuit with each set of simulations performed twice: The first run retained all parasitic parameters, and calculated the deviation of the simulation results from the reference value; the second run retained only the high-sensitivity parasitic parameters, setting all low-sensitivity parameters to zero, and again calculated the deviation from the reference.

The number of Monte Carlo simulations was set to 50. For the core circuit, the absolute relative error in the temperature coefficient between the first and second run is 0.00682. For the op-amp, the absolute relative errors in low-frequency gain,-3 dB bandwidth,Gain-bandwidth product (GBW), Phase margin are 0.00358, 0.00705, 0.00705, 0.00411, respectively.



(a) Temperature drift curve under full simulation



(b) Temperature drift curve simulated only with low-sensitivity parameters

Figure 5: Core circuit temperature drift curves under different simulation strategies

As can be seen, the average absolute relative error in all metrics is less than 0.01, indicating that the low-sensitivity parameters contribute minimally to performance. Thus, eliminating low-sensitivity parasitic parameters still yields reliable simulation outcomes, significantly reducing computational complexity without sacrificing accuracy.

ISSN 2959-6157

6 Conclusion

This paper addresses the impact of parasitic parameters on the performance of analog integrated circuit design, and proposes an automated method for circuit sensitivity analysis and interaction effect evaluation based on Cadence SKILL. The main contributions of this work are summarized as follows:

1. An automated simulation control framework based on SKILL and OCEAN scripting is proposed and implemented which supports multi-parameter batch simulations and data extraction, significantly reducing manual workload and improving the repeatability and efficiency of simulation tasks. The sensitivity analysis and modeling flow developed on this basis can be directly integrated into existing EDA processes, demonstrating strong practical value.

2. To address the large number of potential parasitic parameters in circuits, the Morris method is used to conduct sensitivity analysis on the parasitic elements of interconnects, thereby identifying parameters that significantly impact key circuit performance metrics. This method balances computational efficiency and interpretability of result, laying a solid foundation for subsequent modeling and optimization.

3. The study investigates the interaction effects among parasitic parameters. By designing two-dimensional parameter sweep experiments and implementing a regression analysis model in SKILL, the work quantitatively identifies several parameter pairs that exhibit negative synergistic effects on performance variation. Experimental results confirm that incorporating interaction effect modeling substantially enhances the explanatory power regarding performance trends and provides actionable directions for optimization.

Overall, this work implements a complete workflow from sensitivity analysis to interaction effect evaluation, offering both tool support and theoretical grounding for circuit reliability analysis and parasitic effect modeling.

References

[1]Robert H Havemann and James A Hutchby. High-performance interconnects: An integration overview. Proceedings of the IEEE, 89(5):586–601, 2001.

[2]Max D Morris. Factorial sampling plans for preliminary computational experiments. Tech-nometrics, 33(2):161–174, 1991.

[3] Ilya M Sobol. Sensitivity estimates for nonlinear mathematical models. Math. Model. Comput. Exp., 1(4):407–414, 1993

[4]Antonio J Lopez Martin. Cadence design environment. New Mexico State University, Tutorial paper, 35, 2002.

[5]David Michael King. On the importance of input variables and climate variability to the yield of urban water supply systems. PhD thesis, Victoria University, 2009.