Hierarchical Modeling and Analysis of Process Variations: the First Step Towards Robust Deep Sub-Micron Devices, DC Approach

Pratt School Of Engineering
Duke University

Student: Devaka Viraj Yasaratne
Majors: Electrical & Computer Engineering, Computer Science
Advisor: Prof. Sule Ozev (Assistant Professor of Electrical & Computer Engineering)

Abstract

As feature sizes continue to shrink due to semiconductor manufacturers pushing the limits of process technology, the role of process variations gains more and more importance. As these devices are scaled further down, precise control over process parameters becomes difficult, and knowledge of the variances of these parameters at each hierarchical layer is vital for a successful design process. In my research I collected data to support a hierarchical variance analysis methodology for analog circuits. I performed two different types of analysis in order to determine that this methodology was indeed accurate and computationally efficient, when compared to prior approaches. The two types of analyses were, a Monte Carlo simulation that simulated 50 000 instances of an input stage to an amplifier circuit, and a Sensitivity analyses of that same circuit using first order Taylor series expansions. I then added an output stage in series with the previous circuit, and performed both types of analysis again, to ensure that this approach was effective for larger circuits with more transistors. Experimental results indicate that the proposed first order Sensitivity analysis method is accurate and computationally efficient in circuits that don't have large non-linear dependencies. It is an improvement when compared to previous methodologies when compared to previous methodologies. The research team has drafted a paper titled "Hierarchical variance analysis for analog circuits based on graph modeling & correlation loop tracing" which has been submitted for a blind review.

Table Of Contents

Introduction

As devices have become more and more widely used to perform a multitude of everyday tasks, the need for them to be smaller, faster and more reliable has grown too. In response feature sizes of semi conductor components have shrunk in such a way that device parameters get so small that even the smallest process variations of a few atoms, lead to high variability in the performance. The SIA roadmap for semiconductors! [5] Indicates an expected node size of 65nm (physical gate length = 25nm) in 2007. As these feature sizes shrink, the impact of process variations on the circuit is amplified. As an example, gate oxide thickness has reduced from 140Å for 0.5µm technology to 25Å for 90nm technology. Since the average bond length of SiO2 is 1Å, the minimum absolute deviation in gate oxide thickness cannot be less than 1.61Å. Thus, the minimum relative deviation in oxide thickness has increased from 1.1% for 0.5µm technology to 6.4% for 90nm technology (cite fangs paper!).

These process variations propagate to from the process level to the performance level, leading to variability in the output. If the effect of the said variations could be analyzed and accounted for it would be very useful to design, automation, and test development efforts. During design, analysis of the effects could lead to both better estimations of the circuits yield, and also could be helpful in quickly identifying parts of the circuit where the variation of a process parameter has a large impact on the output. The designer could then quickly change his design before beginning the fabrication process. This type of analysis also leads to the ability to quickly generate tests, which will identify both catastrophic and soft faults, thereby drastically reducing time to production.

It is clear that the need to integrate process variations into the design process is gaining importance and examining a few techniques that have been considered before must be examined. Previously methods for estimating tolerance windows have led to either an under estimation by using arbitrary margins for process variations [fang 5!] or an over estimation by only taking corner cases into account [fang 6!]. All techniques to evaluate the statistical variations in component values are known as methods to solve the statistical tolerance analysis problem, and an abundance of such techniques exist. They fall into three broad categories, which are derivation-based approaches, sample-and-simulate approaches, and worst-case min-max approaches. Out of these methods the most computationally accurate is the derivation method, although it deriving exact functional relations between process and performance parameters can be complex and time consuming. Worst-case min-max analysis is the opposite of the derivation method in the sense that it provides an extremely inaccurate result at a low computational cost. Sample-and simulate based approaches provide a good understanding of tolerance behavior, albeit at a high computational cost [site fang!].

Due to the several reasons that will be discussed in this section, a new method of solving the tolerance analysis problem is needed. Firstly, while devices have been shrinking systems have also become more complicated. Due to this complexity more and more transistors are being placed on a single die, and this has pretty much eliminated any hope of using a purely derivation based method of tolerance analysis. As designs get more complex, and more dependant on process variations, min-max analysis could start to produce results with a percentage variations of hundreds of percentages. Purely sample and simulate methods require a large number of simulations as circuits become more complex, and this leads to a large increase in the cost of those methods. Also, another reason to find a new approach is the need to reduce computation time in order to be able to push designs to market faster. Since the current approaches are either to inaccurate or costly a new method is needed. A new technique should exploit the hierarchical construct of circuits in order to promote design reuse. The reason for this is that a slight change modification in one part of the circuit will not require a re-computation of the overall tolerance information, and thereby greatly reduce evaluation time.

In this paper a proposed [fang's paper!] novel hybrid hierarchical tolerance technique that exploits the advantages of both the derivation-based analysis method and approximate modeling in an attempt to enable accurate and efficient computation of parameter variations. As the number of hierarchical layers increases the accuracy of the computation does not start to decrease dramatically, and the computation time is dramatically less than the 'Monte Carlo' sample and simulate technique that it is compared to. The goal of this technique is to derive the tolerance response in a hierarchical manner.

Hybrid Hierarchical Tolerance Analysis Method

As discussed before the increasing complexity of electronic circuits has led to the requirement of a computationally efficient, hierarchical approach to tolerance analysis. The proposed approach will use simulation based modeling at the transistor level and rely on analytical derivation for modeling at higher levels of hierarchy.

In analog circuits, a small number of transistors constitute building blocks (such as current mirrors, diff-amp pairs, etc.), several building blocks constitute modules (such as an OPAMP), and modules are connected in a network to building the circuits (such as a filter or an ADC). The relations among parameters of the circuit at various levels can be defined either through approximate behavioral models, or through simulator-based models, such as the sensitivity analysis. Thus, there is inherently a hierarchical construct followed during the design process. This hierarchy can be depicted as follows:

$$P_{i^1} = f_{i^1}(P_{1^0}, P_{2^0}, \ldots, P_{NP_0^0})$$

$$P_{i^2} = f_{i^2}(P_{1^1}, P_{2^1}, \ldots, P_{NP_1^1})$$

$$\ldots$$

$$P_{i^r} = f_{i^r}(P_{1^{i-1}}, P_{2^{i-1}}, \ldots, P_{NP_{i-1}^{i-1}})$$

Where P_{i^r} denotes the i^th^ parameter at the hierarchy Level-r, f_{i^r} denotes its functional relation in terms of parameters one level down in the hierarchy and NP_r denotes the number of parameters at the hierarchy Level-(r). [Date Paper!].In [Fang's

Thesis!] the following expression is derived to compute the variance of each parameter at each level of the hierarchy of the circuit.

$$\sigma^2_{P_{j^i}} = \sum_k (S^{P_{j^i}}_{P_{k(i-1)}})^2 \sigma^2_{P_{k(i-1)}} + \sum_{i^{(0)}}^{r^{(0)}} \sum_{i^{(1)},j^{(1)}}^{r^{(1)}} \dots \sum_{i^{(n-2)},j^{(n-2)}}^{r^{(n-2)}} \cdot \sum_{i^{(n-1)} \neq j^{(n-1)}}^{r^{(n-1)}} (S^{P_{i(n)}}_{P_{i(n-1)}} \dots S^{P_{i(1)}}_{P_{i(0)}})(S^{P_{i(n)}}_{P_{j(n-1)}} \dots S^{P_{j(1)}}_{P_{i(0)}}) \sigma^2_{P_{i(0)}}$$

Equation 1

Where,

$$S^{P_{j^k}}_{P_{j(k-1)}} = \frac{\partial P_{j(k)}}{\partial P_{j(k-1)}} \Big|_{(\mu P_{1(k-1)}, \mu P_{1(k-1)}, \dots, \mu P_{1(k-1)})}$$

Equation 2

Represents the first order sensitivity of a level-k parameter to a level-k-1 parameter $P_{i(k-1)}$. In this report I concentrate on level 0,1 and 2 parameters, and have picked a circuit that has only one level 1 DC parameter that affects the DC parameters of level 2, and therefore Equation 1 simplifies to only the first term.

Modeling the Circuit

The circuit used was the 7-transistor operational amplifier circuit depicted in Figure 1. All transistors have a length of $1.8 \times 10^{-7} m$ in order to allow a even a change of a few angstroms to significantly change the input parameter values.

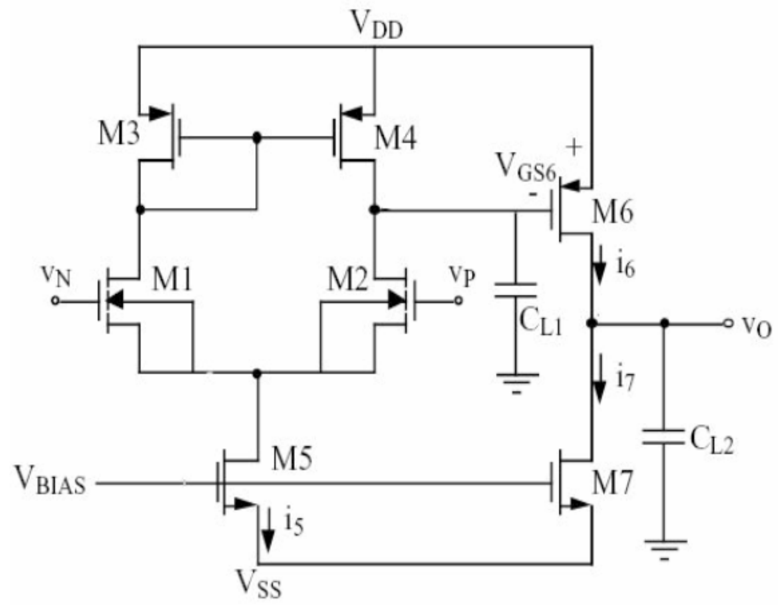

Figure 1

This circuit can be broken into two major blocks, which are connected in a hierarchical manner. The first block would be the first 5 transistors that make up the input stage. This stage, which is depicted in figure 2, is connected to the second stage by the voltage at the node that connects transistors M4 and M2 together. This voltage will be known as V_{out}. The second stage, or top level of this basic hierarchy is the output stage depicted in figure 3. The variances of the output parameters of the second stage (i.e. $I_{d6}, I_{d7}, V_o ...$) are dependent not only on variations of the process parameters of that stage (i.e. $W_6, W_{7,} Tox_6, L_7 ...$), but also on the variance of V_{out}.

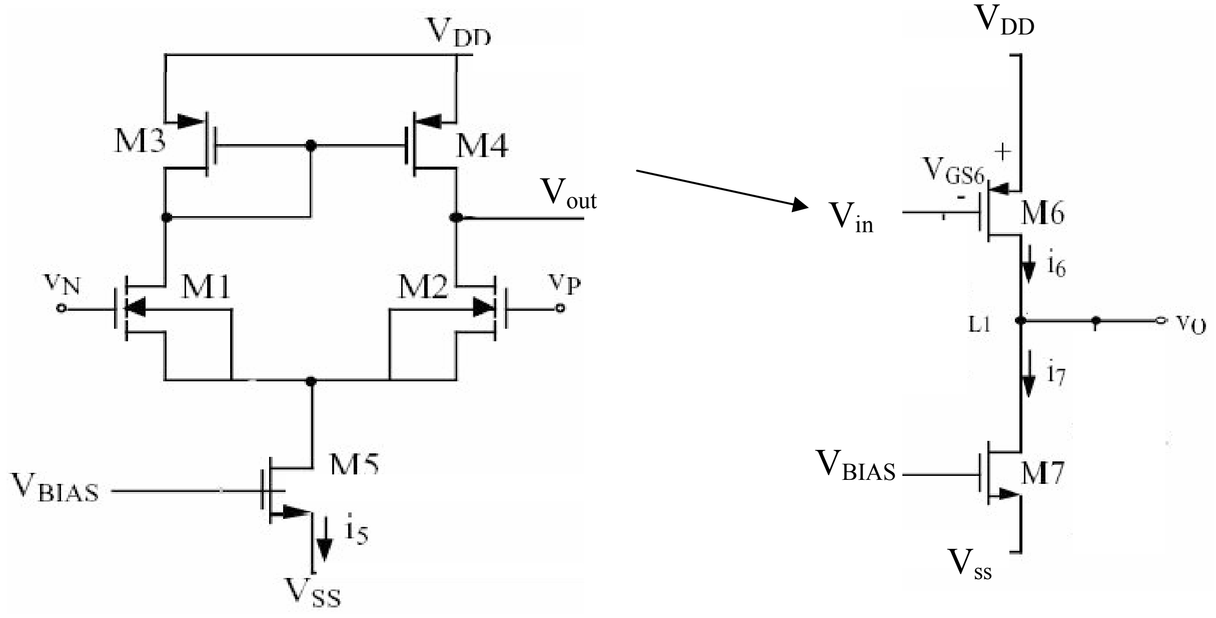

Figure 2 **Figure 3**

The hierarchy can be displayed more clearly by utilizing a graphical approach as shown in figure 4

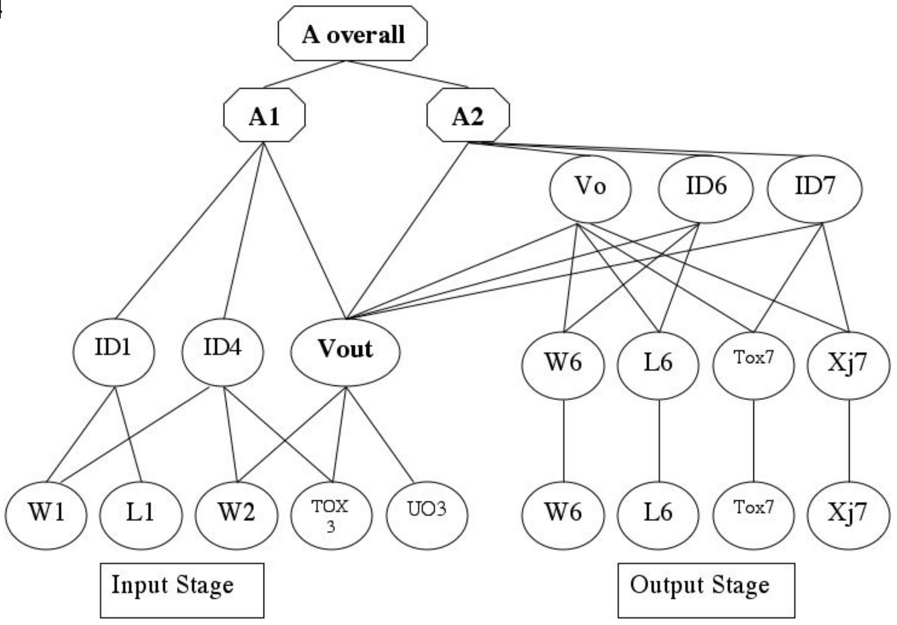

Figure 4

The circuit described above was modeled using a level 49 HSpice circuit file. Nominal values for the input parameters were selected from a circuit that was assumed to be used in industry today. All Sensitivity and Monte Carlo simulations were performed using this file as the basic input method. The HSpice input file can be found in the Appendix section of this report.

Simulation

As was discussed previously a monte carlo simulation was performed to find out what the actual variances of the output parameters were, and then these values were compared to the sensitivity based tolerance analysis described by Equation 1.

Accounting for mismatch

The transistors M1, M2 and M3, M4 have to be matched in order for the circuit to function correctly, keeping all transistors in the saturated region. As these transistors are matched, they will be laid out close to each other on the die, and due to this process variations of corresponding input parameters (i.e. W1,W2) would be similar. Therefore, in order to accurately simulate process variations in the nominal circuit it was necessary to introduce a small mismatch percentage between corresponding input parameters in matched transistors, while at the same time varying the 2 parameters by a greater percentage from its nominal value. With this technique the matched parameters (i.e. W_1, W_2) become a single parameter (i.e. W_{12}), and the mismatch becomes another independent parameter (i.e. mmW_{12}).

Calculating Sensitivities

The method proposed in [Fangs thesis!] to calculate the variance of output parameters is implemented mathematically in Equation 1. In order to perform this computation it is necessary to find out the sensitivity of each upper level parameter to each lower level parameter. The basic calculation of sensitivity is shown in Equation 2. In order to account for the mismatch between matched parameters an additional term needs to be added to Equation 1, and this change is depicted below:

$$\sigma^2_{P_{j^i}} = \sum_k (S^{P_{j^i}}_{P_{k(i-1)}})^2 \sigma^2_{P_{k(i-1)}} + \sum_m (S^{P_{j^i}}_{P_{m(i-1)}})^2 \sigma^2_{P_{m(i-1)}}$$

Equation 3

Where, P_m is the mismatch of 2 parameters, and m is the number of mismatched parameters. A sensitivity simulation was first performed on the first/input stage circuit shown in figure 2. Initially sensitivities were calculated for every single input parameter of the five transistors, by perturbing 1 parameter by 1% of its nominal value and measuring the effect of that change in terms of the percentage change in the following output parameters for each transistor: ID, GM, GM, GDS, CGS, VDS, CGD, CDTOT. After performing this initial analysis it was clear that the output parameters were much more sensitive to certain input parameters. In order to further improve computation time, a list of input parameters that were significantly sensitive to the output parameters was selected, and all further simulations were run on these. These input parameters were W, L, CGSO, CGDO, TOX, VTH0, K1, K2, PDIBLC2, U0, XJ, NFACTOR, PCLM, VOFF, PB, and CJ for each transistor. In order to account for matching a second sensitivity simulation had to be performed on the 5 transistor input stage. In this second simulation all parameters were kept at the nominal value including one of the matched pair (i.e. W_1 in W_{12}). The other parameter (i.e. W_2) was varied by the mismatch percentage of 0.5% and the sensitivity to each output parameter was found. The result of this simulation was then normalized so that it reflected sensitivity to a 1% change in the input mismatches.

After performing 2 sensitivity calculations – to sensitivities to the variation from the mean, and the mismatch variation - on the input stage, a third simulation was run on the output stage. This circuit is depicted in figure 3, and the same pre-selected list of input parameters was once again varied by 1% to find the sensitivities. The output variable V_{out} from the input stage was considered to be an input to this circuit, and was modeled as a voltage source. This is how the hierarchical construct of circuits is exploited in this method of analysis.

Monte Carlo Simulation

A Monte Carlo simulation is a brute force method of finding out how a circuit would realistically be subject to process variations. This simulation was performed by taking the HSpice input file of the circuit in Figure 1 to be the nominal circuit and then normally distributing the list of input parameters specified in the previous section by a 3σ value of 2%, and also normally distributing the mismatch by a 3σ value of 0.2%. 20000 such random circuits were created using the distributions, and they were simulated in HSpice to find the outputs. The results were then analyzed to find the variance of the output parameters that were listed in the previous section.

Coding Infrastructure

In order to perform the simulations described above it was necessary to come up with a coding infrastructure to modify circuit files accordingly, collect data, and automate the process so that these tasks didn't have to be repeated manually for every single HSpice simulation. Since the nominal circuit had to be modified multiple times in both methods of simulation, and a new circuit file need to be generated for each run it was decided to design a new input format that could describe either of the 2 required circuits in about 20 lines. This format is easily converted to files in HSpice format and vice versa. The use of this input format helped to simplify the programming effort, and made it easy to parse through any of the three input circuits in figures 1,2, and 3 with minimal changes to the code.

All the programming was done using Java 1.4.2 developed by Sun Microsystems, and developed on the Eclipse 3.0.0 design environment. The Java programs were run in a Unix environment, and used to perform the necessary modifications, call HSpice, and then collect the results. Java was chosen in order to provide modularity to separate the different simulation steps into classes and also to be able to reduce the code redundancy for the 2 different types of simulations. The classes used and their purpose are described in the following sections:

Classes used for Sensitivity

HSpiceFileGrabber.java – Reads an HSpice input file and translates it into the new easy to parse format. It also stores a ‘y’ or ‘n’ next to each parameter value indicating that it may be varied or not. The list of parameters that can be varied must be specified in this file.

GrabSpice.java - Grabs all the output values of a single simulation and writes it to a specified output file.

CircuitBuilder.java – The percentage that the input parameters must be varied by is the input to this file. A modification must also be made depending on whether the simulation runs are for standard process variation sensitivity calculations or the sensitivity to mismatch calculations. This class goes through the input file created by HspiceFileGrabber.java and builds a new circuit (with one input parameter varied) for each variable input parameter.

HSPiceFileCreatorSens.java - Uses information in myInputFile, and PARAMFILE (which contains the names of all the input parameters) to simulate circuit files created by CircuitBuilder.java. After each HSpice simulation GrabSpice.java is called to grab all the outputs of that run and write it to "o1outputs.temp". After all the simulation runs are complete calcSensitivity.java is called to calculate the sensitivity to each input parameter to all output parameters.

CalcSensitivity.java – This class goes through all the data collected by HspiceFileCreatorSens.java and performs the sensitivity calculation. All sensitivity data is then written to stvt.txt.

$$S_p^o = \left(\frac{O_{New} - O_{Initial}}{O_{Initial}} \right) \cdot \frac{100\%}{\Delta P}$$

Equation 4

Where S_P^O is the sensitivity of output parameter O to a change in input parameter P. This is essentially the same as Equation 3.

calcSensO1.java - calls HspiceFileGrabber.java, CircuitBuilder.java to build all the specific circuits and HspiceFileCreator.java to perform the sensitivity simulations.

Classes used for Monte Carlo

HspiceFileGrabber.java and GrabSpice.java perform the same tasks and are shared by both programs.

MonteCarlo.java - Calls HspiceFileGrabber on the input file, then calls Pickvals.java to generate a specified number of random circuits, and finally calls HSpiceCreator.java to simulate the circuits.

PickVals.java – This file takes in the Number of random circuits to be simulated, and the 3Sigma points of the normal distributions for mismatch and standard variation. It goes through each input parameter in file data.txt and creates NUMCIRCUITS random circuits by varying each of those parameters within the bounds specified by the normal distribution.

HSPiceFileCreator.java - Uses information in myInputFile, and PARAMFILE (which contains the names of all the input parameters) to build and simulate circuit files. After each simulation GrabSpice is called to grab all the outputs of that run and write it to "randout.out". After all the simulation runs are complete GrabParam is called to collect output values of each parameter.

GrabParam.java - grabs all the values of a specified output parameter from a file such as randout.out, which is generated by multiple HSpice simulations.

Experimental Results

The data collected from the 3 sensitivity runs – Sensitivity to standard variation, and mismatch for the input stage, and sensitivity to variations in the output stage – can be found in the Appendix section of this report. After analyzing the data and applying Equation 3 to the V_{Out} (as defined in previous sections this is the output voltage of the input stage) the percentage of the standard deviation from the mean value was found. .Compared to the Monte Carlo Approach the sensitivity method produced a 20% error as can be seen in table 1.

Type of analysis	Standard Deviation of Parameter as a Percentage of the Nominal Value	Error %
Monte Carlo 20K runs	1.585625155	
Sensitivity Method	1.916650749	20.87666135

Table 1: Stdev of output parameter V_{Out} in the input stage

In order to move to the next block in the hierarchy it was necessary to use the connecting parameter V_{Out} as an input to the circuit in figure 3. The sensitivity of V_o and i_{d6} of the second stage to a 1% change in V_{Out} was then manually calculated to be 24.144 and 5.936 respectively. Taking the standard deviation of this parameter to be 1.9167% as was calculated above, and the standard deviation of all the other parameters to be 2/3%(3Sigma point of Monte Carlo run). The following results were obtained:

Type of analysis	Standard Deviation of Parameter as a Percentage of the Nominal Value	Error %
Monte Carlo 20K on V_O	31.6413	
Sensitivity Method on V_O	51.055	61.358
Monte Carlo 20K on I_{D6}	9.0353	
Sensitivity Method on I_{D6}	12.7265	40.8533

Table 2: Stdev of output parameters in the output stage

Since the error seems to have double when propagating up the hierarchy, an extra test was performed to see what happens if Monte Carlo was performed on the 2 blocks separately and then combined using the sensitivity tolerance analysis technique. In order to do this the only change that had to be made was to use 1.5856% as the standard deviation for V_{Out} when it is used as an input parameter to the second stage. The results obtained from this variation are in table 3.

Type of analysis	Standard Deviation of Parameter as a Percentage of the Nominal Value	Error %
Monte Carlo 20K on V_O	31.6413	
Sensitivity Method on V_O	44.3523	40.174
Monte Carlo 20K on I_{D6}	9.0353	
Sensitivity Method on I_{D6}	10.9769	21.489

Table 3: Stdev of output parameters in the 2nd stage(modified)

What is important to note is that while the Monte Carlo simulation needed 20000 circuit simulations and approximately 13 hours of runtime, the 3 sensitivity calculations included simulation of less than 150 circuit simulations and could be performed in under 30 minutes even while making the necessary changes in the code to find sensitivity to either the standard variation, or mismatch.

Conclusions

Two main conclusions can be reached from the research that has been performed. Firstly, the technique proposed to calculate the variance of an output parameter using the sensitivities of this parameter to a list of input parameters as described in Equation 1, seems to under estimate the actual variation by a large percentage (20%). One could argue that since this method is still useful as it is relatively accurate in comparison with something like a min-max analysis, and can also be performed so quickly it still has practical applications such as rapid test development, and easily identifying problem areas in circuits without delay. However, the fact that the error seems to almost double as one moves from one level of the hierarchy to the next means that this approach gets more inaccurate for circuits with large non linear dependencies, such as the dependency of the output parameters of the second stage on V_{Out} of the input stage. More research needs to be done to find methods of adding additional correction terms to the equation in order to make it work better with highly non-linear circuits.

Table 3 shows results acquired when the Monte Carlo analysis is done separately for each block (This is feasible as the Monte Carlo approach for small blocks is less costly), and then calculated sensitivities are used to find the total variance of performance parameters. This method is more costly but almost twice as accurate as the purely sensitivity based technique.

Future Work

Previously, the 2nd term of Equation 1 was ignored, as there were no correlations between lower level parameters. The next step would be to design a circuit with 4 or 5 level parameter hierarchies, and perform this analysis on that circuit. Correlation loops as described in [cite our earlier paper] will come into effect in such a circuit, and an algorithm that identifies such loops needs to be developed.

More analysis of the current technique needs to be done in order to make this proposed approach more accurate.

Acknowledgements

I would like to sincerely thank Prof. Sule Ozev for her guidance, enthusiasm, and patience. I am also extremely grateful to Mr. Fang Liu for introducing me to his work, and Jacob Flomenberg for his assistance in developing the coding infrastructure.

Name: Devaka Viraj Yasaratne
Majors: Electrical and Computer Engineering/Computer Science Double Major
Project: Hierarchical Modeling and Analysis of Process Variations: The First Step Towards Robust Deep Sub-Micron Devices. Nonlinear DC Approach
Advisor: Sule Ozev, Assistant Professor of Electrical and Computer Engineering