Statistical Delay Modeling in Logic Design and Synthesis

Horng-Fei Jyu Sharad Malik
Department of Electrical Engineering
Princeton University, Princeton, NJ

Abstract

Manufacturing disturbances are inevitable in the fabrication of integrated circuits. These disturbances will result in variations in the delay specifications of manufactured circuits. In order to capture the impact of these variations on the delay behavior of these circuits we propose a pair of statistical delay models for use in logic design. These models abstract the real variations from the process level and can be used for statistical delay analysis and optimization in logic design and synthesis while offering an efficiency vs. accuracy tradeoff.

1 Introduction

Manufacturing disturbances result in variations in the delay specifications of the final manufactured circuits. As advanced technology keeps shrinking the size of transistors, this problem becomes more serious. This is because the disturbances do not scale down proportionally with the scaling down of technology as the disturbances are inherently uncontrollable. For example, a $0.25\mu m$ misalignment causes 17% error in the $1.5\mu m$ transistor, but will cause 50% error for $0.5\mu m$ technology. This variation directly translates to variations in the delays of individual gates in the circuit and eventually to variations in the overall delay of the circuit. This increased ratio of unpredictable variations makes it increasingly difficult to be ignored. On the other hand, handling these variations by assuming a worst case bound results in loss of available performance, which translates to loss of competitive edge. This provides the motivation to examine the consideration of these variations during the design of logic circuits.

We propose a statistical delay modeling framework, comprising of two models, where we abstract the effect of these variations as random variables which directly influence the delays of gates in the circuit. We point out the requirements on such a model that come from the delay analysis and optimization stages in logic design, and specifically target our models for ease of use in logic design or synthesis. Between them, the two models offer a range of efficiency and accuracy.

2 Requirements on the Delay Models

There are several driving forces that influence the choice of statistical delay models. The first of these arises from the need to capture the correlations between the different parameters at the device and circuit levels.

2.1 Correlations between Parameters

We start by pointing out the effects of the fabrication variations on the delay of devices and gates. The variations in fabrication can be captured at several different levels. At the logic design level, the unit we want to handle is a gate. Unfortunately, the delay of each gate is not independent of the others. Consequently, we need to examine the source of independent variations. As shown in Table I, there are several different

Table I: Typical CMOS parameters at different levels

Device parameter	Process parameter
V_t : threshold voltage	t_{ox} : thickness of oxide
β : transconductance	W : channel width
C : load capacitance	L : channel length
	$Temp.$: temperature
	N_{sub} : doping density

levels of delay parameters. The delay of a logic gate (d_g), a circuit level parameter, is a function of device level parameters of V_t, β and C_{out}. In turn, these device parameters are not independent, they come from the same source of process parameters such as t_{ox}, W, L and N_{sub}. These process parameters can be considered to be independent primitive parameters and a direct consequence of the fabrication variations. Let us consider the task of simulating the delay behavior of a logic circuit starting with one instance of the process parameters. For an accurate simulation we need to generate all the process parameters independently, giving us a sample $\vec{p_p}$. This process sample $\vec{p_p}$ can be used to derive a specific device parameters vector $\vec{p_d}$ as well as a specific circuit parameters vector $\vec{p_c}$. It is important to note that the components of $\vec{p_d}$ as well as $\vec{p_c}$ are not independent of each other since they depend on the same process parameters. Due to this correlation between them, we need to start the delay simulation from the process parameters. Otherwise, the final result will be inaccurate as it fails to capture these correlations.

Having emphasized the importance of the process parameters, we next examine the environment in which the statistical models will be used and look at the demands imposed by such an environment.

2.2 An Environment for Statistical Delay Optimization

The overall goal of this research is to incorporate the effects of statistical fabrication variations during

logic design and synthesis. This is accomplished by trying to predict the final delay behavior of the manufactured circuit, not in terms of a single number or a set of numbers, as is done in worst-case design; but rather as a delay distribution which reflects the fabrication variations. Such a delay distribution can be obtained by a statistical delay analysis step [6], which uses a statistical delay model for individual gates. In such an environment, the goal of delay optimization is to maximize the number of circuits expected to exceed a certain performance level based on the computed circuit delay distribution. Such a delay optimization step can be automated as demonstrated in [5]. This optimization program uses the analysis stage in the inner loop to evaluate the quality of the designs it is considering. Thus it is important that the analysis step be efficient, since it will be used repeatedly. This in turn puts demands on the statistical delay models used during analysis; they must permit an efficient analysis while at the same time not sacrifice accuracy. In order to better understand the requirements posed by the analysis stage, we present a brief overview of our analysis technique, the details can be found in [6].

This approach comprises of two stages: `analysis stage` and `simulation stage` In the first stage, we find out all the possible critical paths by considering the delay variations. In order to enhance the efficiency of this analysis, we need to prune out those paths which can never possibly be critical. This is illustrated through the example shown in Figure 1. Here delay variables d_1, d_2, represent gate delays, and lie in the range [1,3] and [2,4] respectively. These bounds on the gate delays reflect the fact that the fabrication variations themselves tend to be bounded. In fact, it is precisely these bounds that we exploit to prune out paths that can never be critical.

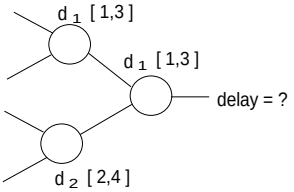

Fig. 1: Statistical delay analysis example

The delay at the circuit output can be expressed as:

$$\begin{aligned} \text{delay} &= max(d_1, d_2) + d_1 \\ &= \begin{cases} 2d_1 & \text{if} \quad d_1 > d_2 \\ d_1 + d_2 & \text{if} \quad d_2 \geq d_1 \end{cases} \end{aligned}$$

Each of these inequalities represents a path in the circuit that is a candidate critical path. The feasibility of each inequality can be evaluated by utilizing the delay bounds of each delay variable. Suppose the delay bound of d_2 changes to [3,5], then the first of these path delays can be pruned out since d_1 can never be greater than d_2. When used iteratively at each node of the circuit graph, this procedure trims down the list of candidate critical paths from all the paths in the circuit to a small set which is then used in the simulation stage in analysis.

After obtaining all the statistically longest delay representations, which are represented by the linear combinations of variables, we then enter the second stage where we run the simulations. A process simulator is used to generate the samples for the delay variables. Since the process parameters are the source of disturbances, and hence are independent, the correct correlations among the variables can be captured. A flow chart of this procedure is shown in Fig. 2.

The advantage of using this analysis technique is that it avoids an expensive monte-carlo simulation of the entire circuit and replaces it with the relatively inexpensive monte-carlo simulation of a few path delay expressions. The success of this technique depends on the ability to prune out a large number of paths that cannot be critical. This is accomplished using linear programming techniques and thus requires that path delay expressions be expressed in a linear form. Thus, the requirement imposed by analysis is that the statistical delay models be such that the path delays can be expressed as a linear sum of delay variables.

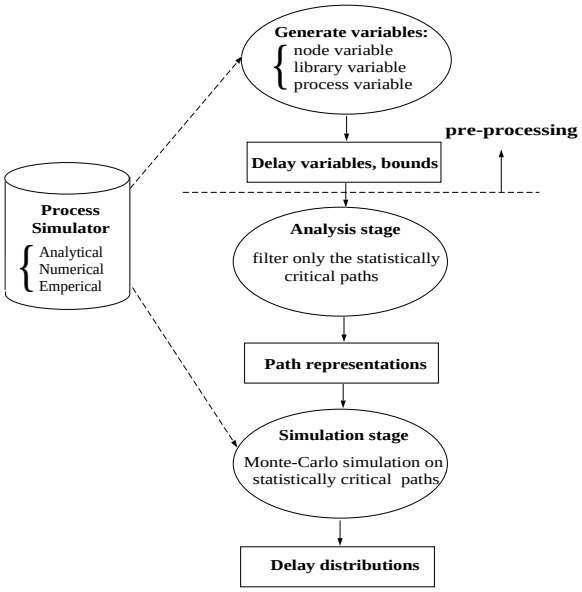

Fig. 2: Flow chart for statistical delay analysis

3 Abstraction of Statistical Variables

The variations of the final delay of an integrated circuit come from the disturbances in the fabrication process. Within a chip, there are two primary variations: global disturbance and local disturbance. Global disturbance is the disturbance which is the same for all the elements inside the chip. Local disturbance is the one specific to each element inside a chip. For example, misalignment of the mask will result in a global disturbance which effects the widths and lengths for all the gates inside the chip. Section 3.1 and Section 3.2 provide two ways to abstract the global disturbance. Section 3.3 will discuss the effects of local disturbances.

3.1 Library Variable

The most straightforward way for delay abstraction in the logic design stage is to represent the delay of a gate (denoted as a node i in the Boolean network) as a

random variable N_i. Therefore the delay of a path P_j can be represented by :

$$delay(P_j) = \sum_{\text{node } i \text{ is on path } P_j} N_i$$

For example, if a path consists of four nodes, with delays N_1, N_2, N_3 and N_4, then the delay representation of this path is expressed as

$$delay = N_1 + N_2 + N_3 + N_4 \quad (1)$$

In this case, the number of variables is equal to the number of nodes in the circuit. We need to reduce the number of variables to improve the efficiency of analysis. By examining the delay formula for a gate, we can see that the delay has two parts: intrinsic delay and fanout delay. Thus, the delay for a node i can be represented as:

$$delay(\text{ node i}) = N_i = A_i \times (C_i + \sum_{k \in fanout(i)} C_k)$$

where A_i represents the driving capability, C_i the intrinsic capacitance and C_k the load capacitance. For standard cell design, all the mapped gates and their fanout loads are selected from a cell library. Therefore, we can use one random variable $L_{i,j}$ to represent one possible combination of $A_i \times C_j$. The range of each variable $L_{i,j}$ can be determined by running the process simulator with the expression for $A_i \times C_j$ described in terms of the process parameters. Then the maximum and minimum values for this AC pair can be obtained. This pair is abstracted as a new variable L_k, and the extracted value from the simulation provides the delay bounds for this new variable. Since a library cell is typically used several times in a circuit, the number of different combinations of $A_i \times C_j$ occurring in the mapped circuit is often much less than the number of nodes. Thus the number of variables can be reduced. Besides, decomposing the delay into common primitives also helps the path pruning process in analysis. For example, if there are two random gate delay variables N_1, N_2, and each of them consists of two components

$$N_1 = L_1 + L_2$$

$$N_2 = L_1 + L_3$$

where L_1, L_2 have a delay range bounded by $[2,3]$ and L_3 is bounded by $[1,2]$. If we abstract the delay variations by variables N_1, N_2, then we cannot tell which is larger when we compare these two delays, since the range for $N_1([4,6])$ and the range for $N_2([3,5])$ overlap. However, if we abstract the delay by L_1, L_2 and L_3, then after excluding the common part of L_1, we can see that the delay of N_1 is always greater than N_2 as a direct consequence of the fact that the delay variable L_2 is always greater than L_3.

With this the path delay is expressed as:

$$delay(path) = \sum_i N_i = \sum_i \sum_j a_j \times (A_{ji} \times C_{ji})$$

$$= \sum_k a_k \times A_k \times C_k = \sum_k a_k \times L_k$$

For the path delay expression in Equation 1, if

$$\begin{aligned} N_1 &= A_1C_1 + A_1C_2 &= L_1 + L_2 \\ N_2 &= A_2C_1 &= L_3 \\ N_3 &= A_1C_2 &= L_2 \\ N_4 &= A_1C_1 &= L_1 \end{aligned}$$

then

$$\begin{aligned} delay &= N_1 + N_2 + N_3 + N_4 \\ &= 2L_1 + 2L_2 + L_3 \end{aligned} \quad (2)$$

Another advantage of this approach is that the library gate variable is independent of the logic circuit, enabling all the required information for the $A \times C$ variables to be pre-computed before the logic circuit is given. We do not need to run process simulations to generate the delay bounds for these variables for every design.

3.2 Process Variable

The motivation for generating library variables is to reduce the number of variables by finding the common primitives in representing the delay information. In fact, all the delay information comes from the process parameters. If we consider only the global disturbance, all the process parameters will have the same value on a single integrated circuit. This can be used to even further reduce the number of random variables. Unfortunately, the delay of a component is usually a very complex function of the process parameters rather than a linear combination of the process parameters, which is imposed by the efficiency requirements of statistical delay analysis. However, if the variations of the process are relatively small, then the delay formula can be approximated by a linear form of the process parameter variables with tolerable errors by utilizing multiple regression techniques [8]. This is a common practice used in device modeling for curve fitting. A similar approach has been adopted in statistical timing verification [1].

Although there are many process parameters, not all of them are used in the multiple regression. The selected parameters to interpret the delay can be predetermined by running the device simulation to see which parameter has significant effects. In Ping Yang's work [10], it has been shown that only four process parameters have significant effects on the delay variations. In general, it can be technology-dependent. As in the case of the library variables, we approximate the delay of a path by

$$t_P = \sum_k a_k \times L_k$$

where

$$L_k = A \times C = f(\Delta W, \Delta L, t_{ox}, N_{sub}) =$$

$$a_0 + a_1 \times \Delta W + a_2 \times \Delta L + a_3 \times t_{ox} + a_4 \times N_{sub} \quad (3)$$

Here, we only consider global disturbances. So N_{sub} and t_{ox} are the same throughout the chip. Different

transistors may have different sizes. However, the variations for all the widths(ΔW) and lengths (ΔL) are the same if we only consider the global disturbance. The advantage of using the linear form is that now we have only the process parameters as variables during the analysis stage and their number in addition to being small, is also independent of the size of the circuit. This permits efficient analysis. To illustrate this with an example, consider the delay in Equation 2. Assume that after multiple regression, we can express the library variables in terms of two process variables, P_1 and P_2 as follows:

$$\begin{aligned} L_1 &= P_1 + 2P_2 \\ L_2 &= 2P_1 + P_2 \\ L_3 &= 0.5P_1 + 2P_2 \end{aligned}$$

then the path delay can be expressed as:

$$\begin{aligned} delay &= N_1 + N_2 + N_3 + N_4 \\ &= 2L_1 + 2L_2 + L_3 \\ &= 6.5P_1 + 8P_2 \end{aligned} \quad (4)$$

Now the various path lengths can be directly compared by comparing their expressions in terms of the two process parameters, making the feasibility check using linear programming comparatively very simple.

Some typical results for the errors versus the variations are shown in Table II. In this experiment, all process parameters are assumed to have uniform distributions. "Sample range" refers to the range for the process parameters used in the derivation of the linear using multiple regression. "Check range" refers to the range from which sample process parameters were generated for use in the linear form. From these results we can see that the linear approximation can get accurate results when the variation range is small. This error, however, becomes more significant ($\simeq$ 10%) when the variation of each parameters increase to 50%. This limitation makes the process variable approach only suitable for small variations.

Table II: Relative errors for multiple regression

Sample range	Check range	Error
[0.85 1.15]	[0.9 1.1]	0.67%
	[0.85 1.15]	0.64%
	[0.8 1.2]	0.99%
[0.7 1.3]	[0.9 1.1]	3.62%
	[0.8 1.2]	2.77%
	[0.7 1.3]	2.60%
	[0.5 1.5]	6.64%
[0.5 1.5]	[0.9 1.1]	11.05%
	[0.7 1.3]	9.06%
	[0.5 1.5]	8.12%

Figure 3 shows the use of the library and process variables as two alternate models in this framework.

3.3 Local Disturbance

Local disturbance can potentially result in large variations [4]. There are three types of local disturbances across the die : random variations, edge effects and striation effects. The latter two cases depend on the final layout, which is beyond the modeling scope at the logic level. For a pre-layout stage of logic design, only random effects can be possibly modeled. These effects account for the unpredictable variations, and therefore have no correlation. Unlike the global disturbance, this variation does not have common factors that can be extracted to reduce the number of variables. Therefore the performance of the analysis stage (Section 2.2) will deteriorate if we include one such variable for each gate in the circuit.

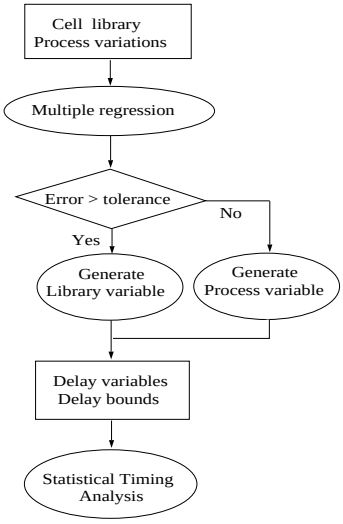

Fig. 3: Pre-processing delay variables

In practice, local disturbances are more significant in analog circuit design, where a perfect match between two elements might be crucial for correct functioning. For our purpose of delay analysis of the complete circuit, the effects of the local disturbances tend to cancel each other. Therefore, local disturbance is intentionally ignored for efficiency [2] [3].

4 Experiments

In this section we describe the experiments conducted to compare the efficiency and accuracy of the two different modeling approaches. The process simulator in Figure 2 has a complicated built-in model based on the real process technology. In the absence of access to any real technology, the process simulator is replaced by simple analytical solutions. The equivalent geometry of the complex gates is derived from a standard cell library `lib2.genlib`.

With these, the analytical solutions for all the gate cells can be built up. They are formulas in terms of the process parameters and serve as input for process simulator. Delay bounds of all library variables L_{ij} for all the cells are calculated for the `library variable` case. Multiple regression is then used to approximate the library variables in terms of the linear combination of process variables. Both these procedures are preprocessing steps conducted once for each library and not once for each circuit and thus can be stored in a "statistical variable library".

In order to compare the efficiency and accuracy of the library and process models, we used each of them for the statistical delay analysis of some benchmark circuits. These results have been shown in Table III. In

Table III: Results for efficiency and accuracy

Circuit	Library variable			Process variable		
	mean	variance	CPU(sec.)	mean	variance	CPU(sec.)
b9	38.99	10.55	20.9	39.63	10.78	0.45
c8	54.10	14.47	9.37	55.03	14.82	0.39
cbp	176.05	39.28	3.43	178.41	40.28	0.40
cbp_kms	100.48	23.59	6.44	101.80	24.15	0.42
cla	147.66	35.59	19.22	149.52	36.42	0.41
ripple	180.67	40.16	2.3	182.37	41.09	0.46
C6288	606.32	140.28	1222.40	615.43	144.11	1.56

these experiments, a 25% disturbance is assumed for each process parameter. The columns under "mean" and "variance" represent the expectation and variance of the delay distribution for that benchmark circuit. "CPU" refers to the CPU time in seconds on a SPARCstation to finish the statistical delay analysis. These results show that process variables can achieve two orders higher efficiency with only a little loss of accuracy compared to the result with using library variables. This is consistent with the results from Table II. From Table II, we can also see that process variable does not perform very well for a larger variation range in the process parameters. For that case, as shown in Figure 3, the library variable is the only choice to get acceptable accuracy for the final delay distribution for the circuit.

5 Relationship to Previous Work

There have been several different research efforts in the area of statistical modeling and statistical design. It is important to clearly set this research in the context of previous results. In the design of high performance circuits, statistical design is required to increase the likelihood of obtaining fast circuits. Statistical design techniques have been well applied at the level of physical design [9]. The typical approach used here is to adjust the controllable parameters during the physical design so as to reduce the sensitivity to the manufacturing disturbances or to maximize the yield according to the variation of parameters. Some approaches for statistical modeling are discussed in [7]. Most of them consider the precision and efficiency at the device level which is not suitable for the logic design of large scale digital circuits.

In contrast to this, we propose statistical delay models which deliver high efficiency, accuracy and flexibility for logic circuit design and synthesis. These models can abstract delay variations to serve as the basis for the statistical delay analysis [6] in logic design. This distribution curve for the circuit delay can help price the chips in a procedure called *speed grading*. An accurate delay distribution is also a necessity for statistical delay optimization [5].

6 Conclusions

In this paper, we have proposed a new framework for statistical delay modeling. In this framework we describe two delay models, a library model and a process based model, for use in statistical delay analysis. Both these models are driven by the need for efficiency during analysis without significant loss of accuracy. We show that for small fabrication variations, the process based model provides for very fast analysis. However, as these variations increase, this efficiency comes at the expense of accuracy. In this case recourse has to be taken to the library delay model which provides the accuracy at the cost of slowing down the analysis.

References

[1] Jacques Benkoski. *Statistical Timing Verification by Formal Signal Interaction Modeling in Multilevel Timing Simulator* . PhD thesis, Carnegie Mellon University, September 1989.

[2] Stephen W. Director, Peter Feldmann, and Kannan Krishna. Optimization of Parametric Yield: A Tutorial. *International Journal of High Speed Electronics*, 3(1), 1992.

[3] Stephen W. Director, Peter Feldmann, and Kannan Krishna. Statistical integrated circuit design. *IEEE Transactions on Solid-State Circuits*, 28(3), March 1993.

[4] Andreas G. Andreou et al. Current mode subthreshold mos circuits for analog VLSI neural systems. *IEEE Transactions on Neural Networks*, 2(2):205–213, March 1991.

[5] Horng-Fei Jyu and Sharad Malik. Statistical Timing Optimization of Combinational Logic Circuits. In *Proceedings of the International Conference on Computer Design*, October 1993.

[6] Horng-Fei Jyu, Sharad Malik, Srinivas Devadas, and Kurt Keutzer. Statistical Timing Analysis of Combinational Logic Circuits. *IEEE Transactions on VLSI Systems*, 1(2), June 1993.

[7] Christopher Michael and Mohammed Ismail. *Statistical Modeling for Computer-Aided Design of MOS VLSI Circuits.* Kluwer Academic Publishers, 1993.

[8] Raymond H. Myers. *Classical and Modern Regression with Application.* PWS-KENT publishing Company, 1990.

[9] A. J. Strojwas. *Statistical Design of Integrated Circuits: IEEE special paper selections of Advances in Circuits and Systems.* IEEE press, 1987.

[10] Ping Yang, Dale E. Hocevar, Paul F. Cox, Charles Machala, and Pallab K. Chatterjee. An Integrated and Efficient Approach for MOS VLSI Statistical Circuit Design. *IEEE Transactions on Computer-aided design*, CAD-5(1), Jan. 1986.