Fast Eye Diagram Analysis for High-Speed CMOS Circuits

Seyed Nematollah Ahmadyan, Chenjie Gu, Suriyaprakash Natarajan, Eli Chiprout and Shobha Vasudevan

Abstract—We present an efficient technique for analyzing eye diagrams of high speed CMOS circuits in the presence of non-idealities like noise and jitter. Our method involves geometric manipulations of the eye diagram topology to find area within the eye contours. We introduce random tree based simulations as an approach to computing the desired area. We typically show $20 \times$ speedup in generating the eye diagram as compared to the state-of-the-art Monte Carlo simulation based eye diagram analysis. For the same number of samples, Monte Carlo produces an eye diagram that is $8.51\%$ smaller than the ideal eye diagram. We generate an eye diagram that is $53.52\%$ smaller than the ideal eye, showing a $47\%$ improvement in quality.

Index Terms—Eye diagram analysis, Random tree optimization, Signal Integrity, Nonlinear analog circuits

I. INTRODUCTION

Signal integrity is the major bottleneck to the system’s performance in high speed CMOS circuit. Eye diagrams [3] are the main diagnostic technique for evaluating the signal integrity. Important signal properties such as noise margin and jitter can be measured from the eye diagram using Monte Carlo transient simulations [11], statistical methods [6][4], and analytical convolution-based techniques [1][12].

Transient circuit simulation using Monte Carlo simulations is the most commonly used eye diagram analysis technique for nonlinear time-variant circuits [11]. However, Monte Carlo simulations can take very long (between days to weeks) [11] to fully analyze variations in the channel and circuits. Their coverage of simulation corners is also not as high as desired. Statistical [6][4] and convolution-based analytical methods [1][12] are fast and high coverage, but their scope is limited to linear time-invariant circuits. Also, they produce the final eye diagram contour, but not the corresponding input simulation trace. We present a simulation based eye diagram analysis technique as an alternative to Monte Carlo based methods. We analyze nonlinear time-variant circuits such as CMOS circuits. We argue for the higher coverage of simulation corners using our method as compared to Monte Carlo in the same time. Put differently, we produce the same quality eye as Monte Carlo in up to $20 \times$ lesser time. We also provide the input traces for an eye.

Our technique is two-fold: First, we use geometry to model the eye diagram as an optimization problem. The eye diagram of the circuit corresponds to the minimum of the objective function in our optimization problem. Secondly, we introduce a simulation-based random tree to minimize the objective function. The random tree algorithm determines the input to the circuit and accordingly simulates the circuit to compute the eye diagram.

In current practice, the eye diagram is used as an output. We use the eye diagram itself to compute the worst case behavior of the design. If the eye diagram was representing non-ideal signal behavior, its contours would be distorted, depicting a noisy signal. Our method exploits this relationship by reversing the order and distorting the eye diagram itself. We model the distortion of the eye diagram for parameters such as noise margin and jitter as a distortion functional of that parameter. For example, The noise margin distortion functional models the area inside the contours of the eye diagram. We define jitter and overshoot/undershoot distortion functionals as well. We model these functionals such that an objective function comprising their weighted sum can optimize for the worst case eye diagram.

The sampling based optimization approach we use is based on random trees. We use the random tree algorithm to optimize the objective function and determine the contours of the eye diagram for the given input sequence. Using random tree simulation, we avoid repetitive exploration of the same regions, which is a known problem in Monte Carlo simulations [11]. Furthermore, we provide a better coverage of simulation corners than Monte Carlo transient simulations. These reasons make the random tree more attractive as an optimizing option than other standard optimization algorithms. Much of our efficiency and scalability results from the choice of the random tree as an optimization tool.

There are two circuit inputs to our method. The first input is a deterministic logical input bit pattern to the circuit. The second input is a set of nondeterministic perturbation parameters that model variations, uncertainty in modeling and noise such as voltage fluctuation, input noise and signal timing variations. We model the perturbation parameters as truncated Gaussian random processes. We generate the eye diagram for the pre-determined input bit-sequence. We assume truncated Gaussian distribution for all the perturbation parameters. We automatically determine and cover the corner cases for each perturbation parameter. Finally, we generate the eye diagram corresponding to the selected bit pattern for the worst-case corner of all the perturbation parameters. Using our method, we can, with high accuracy, generate the absolute worst-case eye diagram of the circuit.

Our geometric approach of manipulating the eye diagram using integrals and optimization has many benefits. Our approach is quantifiable and precise with an optima. Our formulation is very efficient and does not impose any significant computational overhead because it can be computed and updated incrementally at every iteration of the algorithm. Our algorithm is adaptable to different scenarios by adjusting the perturbation parameters for computing area in the eye, as well as optimization objectives. The random tree algorithm is simulation based and we can compute the eye diagram of nonlinear time-variant analog circuits including high-speed CMOS circuits.

We use a post-layout CMOS inverter circuit as a proof of concept. To produce the same eye diagram, our random tree algorithm utilizes samples more efficiently and requires $20.66 \times$ less number of samples and $20.14 \times$ less absolutely time. Alternatively, if we execute both Monte Carlo and

This work was supported in part by National Science Foundation (NSF) under Grant 1-483534-239012-191100.
random tree for the same amount of samples, our algorithm provides better coverage of the simulation corners while only imposing 1% absolute runtime overhead in comparison to the Monte Carlo. Finally, we demonstrate the scalability of our algorithm by computing the worst-case eye diagram of a post-layout 7-stage CMOS ring oscillator circuit. We added 35 variation parameters to this circuit, making the input space have 35 dimensions, while the state space has 210 dimensions.

Our contributions in this work are as follows. We present an efficient method for eye diagram analysis of nonlinear analog circuits. We geometrically model the worst case eye diagram as an area under the eye contours. We introduce a random tree algorithm to optimize the distortion functionals for parameters like noise margin, jitter, etc. We demonstrate how a random tree approach is best suited for this optimization problem. We show that our methodology provides a much higher quality eye than Monte Carlo, while being time efficient and scalable.

II. PRELIMINARIES: THE EYE DIAGRAM

An eye diagram [3] is a two-dimensional plot generated by repeatedly sampling and superimposing a signal (Figure 1). Let $V_{OL}$ denote the logic level 0 and $V_{OH}$ denote the logic level 1. The eye diagram consists of samples corresponding to the signal value 0, signal value 1, transition from 0 → 1 and transition from 1 → 0. Important signal features such as noise margin, peak distortion such as voltage overshoot/undershoot and jitter can be measured from the eye diagram. The noise margin denotes the height of the eye, $V_{OH}$ to $V_{OL}$ at peak to peak, which determines the amount of the additive noise at the output. Peak distortion is the amount of the noise as overshoot and undershoot on $V_{OH}$ and $V_{OL}$ voltages. The jitter is determined by the width of the eye.

III. OUR APPROACH FOR EYE DIAGRAM ANALYSIS

We generate the initial eye diagram for any pre-determined input bit sequence. The input bit sequence is defined by the user and can be arbitrarily long. We make a simplifying assumption that the elements of the bit sequence are independent and there is no interdependence between symbols. On the other hand, the focus of this paper is on the worst-case corners in perturbation random processes that effects the input signal such as voltage fluctuations, noise, timing variations, etc. We focus on transient variations in power, input and timing variations such as jitter and rise time and fall time.

We analyze the eye diagram. For each simulation corner, we determine the worst case input that generates the eye diagram with minimum noise margin, maximum jitter, etc. Our methodology (Figure 2) consists of two important phases: i) Measuring the eye diagram using distortion functionals such as noise margin and jitter functional (Section IV), and ii) Using random tree optimization to minimize the distortion functional (Section V). For the given perturbation input corner, the eye diagram with minimum distortion functional corresponds to the eye diagram of the circuit. For the given corner, worst case input determined by our algorithm corresponds to the eye diagram of the circuit.

IV. GEOMETRIC MEASUREMENT OF THE EYE DIAGRAM

In this section we model the eye diagram analysis as a multi-objective optimization problem.

A. Geometric measurement of the eye output

We propose a formulation for the the eye diagram analysis problem. We maximize the signal envelope of the eye diagram. Equivalently, we can minimize the eye closure, i.e. area inside the eye diagram contour. The eye closure determines various signal integrity parameters such as noise margin, jitter and voltage overshoot/undershoot.

Let $\{b_0, \ldots, b_n\}$ denote the n-bit input bit sequence for the circuit. Let $\{Y_0, \ldots, Y_p\}$ denote the p perturbation random processes. Each random process $Y_i$ follows a truncated Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i)$. Let $\{a_1, \ldots, a_p\}$ denote the maximum distance of the perturbation samples from the mean of the distribution. For example, if we want to determine an eye diagram of an inverter circuit where the input bit sequence is 00110. There could be a voltage fluctuation on input signal $Y_1$, where $Y_1$ follows a Normal distribution $\mathcal{N}(0, 0.05^2)$ and the input voltage can deviate up to $a_1 = 6\sigma = 0.3^v$ in that input corner.

Let $v$ denote the output signal of the circuit. Let $w$ denote the window size of the eye diagram analysis where $w = 2 \times T$ where $T$ is the period of the signal $v$. Let $s$ denote the set of signal samples in the eye diagram. Each sample is a pair of voltage and time, denoted by $(s(t))$ and $(s(t))$, respectively. In order to analyze the eye diagram, we decompose the eye diagram into higher and lower eyelids.

Definition 1: The higher eyelid is the set of signal samples corresponding to $1 \rightarrow 1$ and $0 \rightarrow 1 \rightarrow 0$ transitions in the eye diagram (Figure 3a). Similarly, the lower eyelid corresponds to $0 \rightarrow 0$ and $1 \rightarrow 0 \rightarrow 1$ transitions.

We define important eye diagram specifications such as noise margin, jitter and voltage overshoot/undershoot w.r.t. the signal envelope of each eyelid.

Definition 2: The frontier set is the signal envelope of the lower and higher eyelid.

Noise margin functionals: We measure the noise margin using the integral of the eye diagram contour of the minimum of the higher eyelid (As shown in Figure 3b) and maximum of lower eyelid w.r.t. the time. The minimum of higher eyelid corresponds a weak logical 1 and maximum of lower eyelid corresponds to a weak logical 0. These integrals denote the area within the intersection of higher and lower eyelids. Minimizing this area as a result of minimizing these integrals results in lower noise margin.

The noise margin functionals measure the area inside the higher and lower eyelids. We define these functionals in such a way that minimizing them results in lower noise margin in the eye diagram. Let $s_1(t)$ and $s_2(t)$ denote the minimum higher eyelid and maximum lower eyelid at the time $t$, respectively.
Similarly, the second period respectively. The states with the maximum and minimum time annotation (rightmost and left-most samples) in the higher eyelids in the first and second period respectively. The jitter distortion functionals \( g_3 \) and \( g_4 \) are defined as:

\[
\begin{align*}
  g_3 &= \int_{V_{OL}}^{V_{OH}} s_3(v) - s_3^{Utopian}(v) dv \\
  g_4 &= \int_{V_{OL}}^{V_{OH}} s_4(v) - (W - s_4^{Utopian}(v)) dv
\end{align*}
\]

where \( s_3^{Utopian}(v) \) and \( s_4^{Utopian}(v) \) denote the rise time and fall time of the signal, respectively. Figure 3c shows the result of the jitter distortion functional \( g_3 \) where \( s_3^{Utopian} = 40 \text{ps} \). Similarly, \( g_5 \) and \( g_6 \) are defined for the lower eyelid as well.

We define noise margin distortion functional as:

\[
\begin{align*}
  g_1 &= \int_0^w \left( s_1(t) - s_1^{Utopian}(t) \right) dt \\
  g_2 &= \int_0^w \left( s_2^{Utopian}(t) - s_2(t) \right) dt
\end{align*}
\]

where the Utopian functions \( s_1^{Utopian}(t) \) and \( s_2^{Utopian}(t) \) denotes the minimum and maximum for output voltages and \( w \) is the time window of the eye diagram.

**Jitter**, unlike noise margin or overshoot/undershoot, is a mapping from voltage to time. Although minimizing noise margin integrals also minimizes jitter as well, we emphasize on jitter performance in high-speed IO circuits separately. We measure jitter using the Lebesgue integral [2] of the frontier set from \( V_{OL} \) to \( V_{OH} \) w.r.t voltage as shown in Figure 3c. We define these functionals in such a way that minimizing them results in higher jitter in the eye diagram. Let \( s_3(v) \) and \( s_4(v) \) denote the states with the maximum and minimum time annotation (rightmost and left-most samples) in the higher eyelid in the first and second period respectively. The jitter distortion functionals \( g_3 \) and \( g_4 \) are defined as:

\[
\begin{align*}
  g_3 &= \int_{V_{OL}}^{V_{OH}} s_3(v) - s_3^{Utopian}(v) dv \\
  g_4 &= \int_{V_{OL}}^{V_{OH}} s_4(v) - (W - s_4^{Utopian}(v)) dv
\end{align*}
\]

It is noted that \( s_3^{Utopian} \) and \( s_4^{Utopian} \) are defined for the lower eyelid as well.

**Overshoot and undershoot** are computed using the integral of maximum of higher eyelid and minimum of lower eyelid using the area outside the higher and lower eyelid curves. Minimizing these integrals increases the maximum of higher eyelid and minimum of lower eyelid and results in maximum overshoot and undershoot, respectively. Let \( s_7 \) and \( s_8 \) denote the set of maximum higher eyelid and minimum lower eyelid samples of the signal. The overshoot and undershoot distortion functionals are defined as:

\[
\begin{align*}
  g_7 &= \int_0^w \left( s_7^{Utopian}(t) - s_7(t) \right) dt \\
  g_8 &= \int_0^w \left( s_8(t) - s_8^{Utopian}(t) \right) dt
\end{align*}
\]

where \( s_7^{Utopian}(t) \) and \( s_8^{Utopian}(t) \) are some values that the signal will surely never reach. For CMOS digital circuits, we use \( s_7^{Utopian}(t) = 1.2V \) and \( s_8^{Utopian}(t) = -0.2V \).

**B. Computing the worst-case corner for the eye diagram**

We define the eye closure functional as:

\[
\begin{align*}
  g_1(\{Y_0, \ldots, Y_p\}) &= \sum_{i=1}^{8} \omega_i g_i(x(\{b_0, \ldots, b_n\}, \{Y_0, \ldots, Y_p\}))
\end{align*}
\]

where \( x(\{b_0, \ldots, b_n\}, \{Y_0, \ldots, Y_p\}) \) is a sample in the eye diagram. \( \{b_0, \ldots, b_n\} \) is the input bit sequence specified by the user. \( \{Y_0, \ldots, Y_p\} \) is the perturbation random processes. The weights \( \omega_1, \ldots, \omega_8 \) are defined by the user s.t. \( \sum \omega_i = 1 \) and specify the importance of each distortion functional in the shape of the eye diagram. We want to minimize the eye closure. To minimize \( g_1 \), we have to minimize each distortion functional \( g_i \). The eye diagram with minimum \( g_1 \) corresponds to the eye diagram of the circuit. Since bit sequence \( b_i \) is selected.
by the user, our objective is to find the random processes $Y_i$ that results in the minimum $g$ and the eye diagram of the circuit.

To the best of our knowledge, there is no direct analytical method to optimize or even solve the objective function $g$ [7]. We thereby use a simulation based optimization approach that provides a close approximation to the eye diagram of the circuit.

V. MINIMIZING DISTORTION FUNCTIONALS USING RANDOM TREES

A. The random tree algorithm

We use a random tree (Figure 4) to simulate the circuit. The tree is incrementally grown by adding an edge between an existing node and a new state. Each node is a point from the state space of the analog circuit. Each edge is a short SPICE simulation of the circuit with a specific input trajectory. At each iteration, we select a node $q_{from}$ where we wish to branch. To determine which input trajectory to take, we randomly shoot multiple trajectories from $q_{from}$ in order to determine an optimum trajectory of the circuit at $q_{from}$. We provide details of how we compute the optimum trajectory in the next section. Next, we select the optimum trajectory and simulate the circuit from $q_{from}$ to get the new node $q_{new}$. Finally the tree is expanded from $q_{from}$ to $q_{new}$.

B. Our algorithm to minimize distortion functionals

We use random tree algorithm to minimize the distortion functionals and obtain the eye diagram with minimum eye closure (Algorithm 1). Initially, we apply an initial bit pattern to the inputs of the circuit to exercise the initial eye diagram. At every iteration, we choose which distortion functional we wish to minimize from $g_1, \ldots, g_8$ with probability $\omega_i$ (Equation 4). The random tree algorithm simulates the circuit and samples perturbation inputs that reduces the distortion functionals. This process continues until we converge to the eye diagram of the circuit.

At every iteration, let $g_i$ denote the distortion functional that we wish to minimize. The random tree algorithm randomly picks the node $q_{from}$ from the frontier set $s_i$ of the distortion functional $g_i$.

After selecting $q_{from} \in s_i$, we compute the perturbation input that decreases the distortion functional $g_i$. The distribution and amplitude of the perturbation is defined by the user. We sample a finite number of trajectories from $q_{from}$ by linearizing the circuit at $q_{from}$ and computing the optimum trajectory from the Jacobian matrix [9]. We pick a trajectory $y_0, \ldots, y_p$ that minimizes the distortion functional $g_i$. We simulate the circuit from the node $q_{from}$ for time $\Delta t$ using the input trajectory $y_0, \ldots, y_p$ to get the new node $q_{new}$. The user defines the simulation step $\Delta t$. Finally we add the new node $q_{new}$ to the random tree and update the eye diagram.

We terminate the algorithm after reaching the maximum number of iterations. We also terminate if we converge to the final eye diagram where the consecutive change in the value of distortion functionals is below the threshold.

Algorithm 1 Our algorithm for minimizing the distortion functionals

1: **Input:** bit sequence $b_0 \ldots b_n$
2: **Input:** perturbation random processes $\{Y_0, \ldots, Y_p\}$
3: $G$.init()
4: Initial eye diagram $I$ = Simulate the circuit with input bit sequence $b_0 \ldots b_n$
5: while terminating condition not met do
6: $g_i$ = Select objective with probability $\omega_i$
7: $s_i$ = Select frontier set of $g_i$ from $I$
8: $q_{from}$ = Select a node randomly from the set $s_i$
9: $\{y_0, \ldots, y_p\}$ = Find an optimum trajectory $y_0, \ldots, y_p$ from random processes $\{Y_0, \ldots, Y_p\}$ from $q_{from}$ that reduces $g_i$
10: $q_{new}$ = simulate the circuit from $q_{from}$ using input trajectory $\{y_0, \ldots, y_p\}$
11: $G$.expand($q_{new}$)
12: Update the eye diagram $I$ and its frontier sets
13: end while

The output of the random tree algorithm is the eye diagram of the circuit, parameters for the worst case IO excitation, and the input sequence (bit pattern and variation in parameters such as noise and voltage fluctuations) corresponding to the output eye diagram.

VI. EXPERIMENTAL RESULTS AND DISCUSSIONS

In order to evaluate our algorithm, We implemented a tool in C++ and developed the interface with Synopsys HSPICE for simulating analog circuits. Our experiments were performed on a Core-i52500K processor equipped with 16GB memory.

A. Efficiency of random tree algorithm

We use a post-layout CMOS inverter circuit (Figure 5) to evaluate the efficiency of random tree algorithm vs Monte Carlo transient simulations for computing the eye diagram of the circuits. The inverter has 31 dimensions. The input to the circuit is a binary signal with non-ideal rise and fall time and input jitter. There are 5 variation sources in this circuit on the circuits. The inverter has 31 dimensions. The input to the circuit is a binary signal with non-ideal rise and fall time and input jitter. There are 5 variation sources in this circuit on the circuits. The inverter has 31 dimensions. The input to the circuit is a binary signal with non-ideal rise and fall time and input jitter. There are 5 variation sources in this circuit on the circuits. The inverter has 31 dimensions. The input to the circuit is a binary signal with non-ideal rise and fall time and input jitter. There are 5 variation sources in this circuit on the circuits. The inverter has 31 dimensions.

Efficiency: Figure 6a shows the eye diagram of the inverter circuit obtained using Monte Carlo transient simulation for 50,000 iterations. Each iteration is a small simulation step for
Fig. 6: The worst-case analysis of the eye diagram in Monte Carlo vs our algorithm. Given the same number of iterations, our algorithm generates an eye diagram that is 47% smaller than the eye diagram generated using Monte Carlo simulation.

Figure 6b shown the eye diagram obtained using our algorithm. We run the random tree algorithm for the same number of iterations as Monte Carlo (50,000). Perturbation parameters were sampled from Gaussian distributions. In our algorithm we set the simulation corner to $6-\sigma$ deviation from the mean of the distribution. For example, input timing variation (jitter) follows a $\mathcal{N}(5\text{ps}, 1\text{ps})$ Gaussian distribution, but can take a value from the range of $[0, 5 + 6 \times 1\text{ps}]$. The probability of a sample with $6\sigma$ deviation in Monte Carlo is 0.00034%, but our algorithm was able to quickly find such corners.

As a result our algorithm was much faster and more efficient than the Monte Carlo. Given the same number of iterations, our algorithm produces a more accurate eye diagram and converges faster than Monte Carlo. In terms of absolute runtime, our algorithm does not impose any significant computational overhead. In our tool, the runtime of the Monte Carlo for 50,000 iterations was 141 minutes whereas random tree took 143 minutes$^2$ which shows only 1% runtime overhead.

Figure 7 shows the progress of our algorithm vs Monte Carlo in each iteration. At every iteration, we reported the size of the eye diagram using Equation 4. Using Monte Carlo, the objective size decreased quickly at the beginning of the simulation, but the rate of convergence slowed down very quickly after a few bits. The random tree algorithm, on the other hand, rapidly converged to a smaller eye closure.

Figure 8 shows the eye diagram contour for different maximum deviations from the means of random processes. We executed out tool for different maximum distance from the mean of the distribution for every perturbation parameter. We plotted the contour of the generated eye diagram for 1$\sigma$ distance to 6$\sigma$ distance. As shown in the figure, as we increase the distance from 1$\sigma$ to 6$\sigma$, the enemy closure becomes smaller and the signal integrity declines.

Finally, we extract input stimuli for generating the eye diagram. Statistical methods and analytical convolution-based methods are unable to do this. Figure 9 shows the scatter plot of the input stimuli for power voltage $VDD$ that generates the logical 1 in our eye diagram. Each input stimulus was a path from the root of the tree to a node in the frontier set $s_1$ corresponding to the minimum of higher eyelid.

$^2$In our implementation, at every iteration in both Monte Carlo and random tree, we had to execute the HSPICE software as an external tool which takes an substantially long time for license checkout.
output sequence of the eye diagram was 11001. Most tests in Figure 9 initially follow the ideal path (the initial eye diagram in Figure 2) which is highlighted in the Figure 9 at voltage 0.9V. Figure 9.a shows the histogram of the VDD inputs in the input sequences. We removed the ideal paths from the histogram (where $VDD = 0.9V$) for clarity. Most of the samples for generating a weak logic 1 came from the tail of the voltage distribution at 0.7V (In Monte Carlo, this distribution was $\mathcal{N}(0.9V, 0.05V)$). Figure 9.b shows the scatter plot of the input stimuli extracted from the random tree. There were total of 130 input sequence corresponding to the minimum of higher eyelid $\frac{\text{window-size}}{\text{1ps}} = \frac{130}{130} = 130$. Figure 9.b shows that the worst-case higher eyelid consists of three separate part. In each part, the signal followed the ideal path for some time and then diverged into that part. The worst case higher eyelid occurred during the $1 \to 1$, $1 \to 0$ and $0 \to 1$ transition. However, most of the samples (including the samples determining the noise margin at $t = 60\text{ps}$) were from the $0 \to 1$ transitions. This information can be used for debugging and validating the circuit.

![Fig. 9: The scatter plot of the VDD inputs for generating the frontier set $s_1$. The left side figure shows the histogram of the VDD inputs samples (we excluded the samples from the ideal path). The right side is the scatter plot of input stimuli drawn over time, which identifies three separate component in the worst-case eye diagram.](image)

B. Scalability of our algorithm

The random tree simulation, similar to Monte Carlo, is highly scalable and can be used on industrial circuits. We analyzed the 7-stage post-layout CMOS ring oscillator circuit in 45nm process to demonstrate scalability. The ring oscillator consists of an odd-number of CMOS inverters (Figure 5) arranged in a ring architecture. As a result, the circuit was unstable and oscillated as expected. We added 35 variation parameters to the circuit and analyzed the eye diagram for worst-case at the output. The state space of the circuit was 210 dimensions and the input space was 35 dimensions. Unlike the inverter circuit, the ring oscillator did not have any digital input signal, so we included the input jitter and rise/fall time model in the input space. Furthermore, the output oscillates so there is no $1 \to 1$ and $0 \to 0$ transitions in the eye diagram. As a result, we didn’t model the overshoot and undershoot functionals ($g_7$ and $g_8$) in the objective function by setting $\omega_7 = \omega_8 = 0$.

![Fig. 10: The eye diagram of ring oscillator circuit computed using our technique.](image)

The random tree simulation, similar to Monte Carlo, is highly scalable and can be used on industrial circuits. We analyzed the 7-stage post-layout CMOS ring oscillator circuit in 45nm process to demonstrate scalability. The ring oscillator

VII. RELATED WORK AND CONCLUSION

The de facto method for computing the eye diagrams is the Monte Carlo transient simulations [11][9][8][5]. However Monte Carlo is too time consuming and does not properly cover the simulation corners with high deviations. Researchers have worked on replacing transient simulations with convolution-based analytical methods [12][1]. Analytical methods provide deterministic eye diagram, but are only applicable to the linear time-invariant systems. In [10], the authors construct the output waveforms using multiple edge responses. On the other hand, statistical eye diagram analysis tools use statistical techniques to determine the eye diagram [6][4].

In conclusion, we present a technique for efficiently and accurately compute the eye diagrams of nonlinear analog circuits.

REFERENCES