Using LCSS Algorithm for Circuit Level Verification of Analog Designs

Rajeev Narayanan, Alaeddine Daghar, Mohamed Zaki, and Sofiène Tahar

Department of Electrical and Computer Engineering,
Concordia University, Montreal, Canada
{r_naraya, daghar, mzaki, tahar}@ece.concordia.ca

Technical Report

Feb, 2012

Abstract

The quality of an analog circuit can be determined by the quality of its output simulation trace(s). Traditional assertion/statistical methods are an efficient quantitative approaches that can detect violation during the simulation, but cannot address the quality of the verified circuit. To have a robust verification framework, it is necessary to complement the quantitative approach with techniques that can perform an automatic estimation of the circuit quality based on its output trace. This report relies on the longest closest subsequence (LCSS), a variant of the longest common subsequence (LCS) to account for process variation and mismatch in analog circuits. At circuit level, the effect of mismatch and process variation that result in offsets (vertical and horizontal) are analyzed by performing parametric and statistical techniques and then applying LCSS to estimate the probability of closest matching. The acceptance/rejection of a circuit is done using bounded hypothesis testing. The approach is illustrated on a Rambus Ring Oscillator circuits for a *90nm* fabrication processes. Advantages of the proposed methods are robustness and flexibility to account for wide range of variations.

1 Introduction

Over the last decade, advanced technologies have allowed designers to develop smaller, faster, low power analog/digital/RF designs in a single system-on-a-chip (SoC). As SoC complexity continuously escalate against the backdrop of aggressive time-to-market schedules, the joint effects of physical (e.g., *process variation*) and environment (e.g., *temperature*) constraints have been a major concern for analog designers, as it has left the circuit vulnerable to noise and offsets [11]. The success of an analog circuit from concept to silicon is measured not only on "how fast it is been designed", but also on "how reliable the design is"? which depends heavily on the accuracy of the models used and the verification environment.

Traditionally, simulation by far is the standard technique to verify analog designs [3]. The analog design flow starts with the schematic capture of the circuit with hand crafted models for active/passive elements. This is followed by netlist extraction and manual verification using a circuit simulator, usually SPICE [16]. Verification involves a number of repeated simulation runs that are exercised by specific/random set of inputs to validate the expected output. Unfortunately, the lack of sophistication in computer-aided design (CAD) tools is starting to overwhelm designers as the verification may take weeks of labor intensive simulation to validate the specification.

To address the issue of simulation run-time, designers have looked at modeling techniques at higher level of abstraction using hardware description languages (HDLs), such that verification can be automated and performed much faster [13]. Of course, this speed-up does not come without a price. The first cost is the accuracy of the behavioral model against the actual transistor-level designs. Secondly, the model has to account for physical and functional constraints [13]. Verification of analog designs are faced with immense challenges with the uncertainties due to unwanted deviation either *vertically* (*DC* offset) or *horizontally* in a signal trace because of component mismatch and process variations.

The *DC* offset is very prominent in differential amplifier circuit and in general, modeled by a series/shunt input offset voltage/current source [6]. Since *DC* offset poses a major threat to the resolution of the analog signal, effort to nullify the effect using the offset cancelation circuit has remain popular among analog designers [18]. Though, the offset analysis is done at circuit level, developing a methodology that could verify an analog circuit with offset condition at a higher level of abstraction has not yet been addressed.

On the other hand, *horizontal* offset common in oscillators, can be considered to be a complete drift in the time axis, called as *start-up* delay or a shift in the signal frequency, referred as *frequency* offset [6]. As oscillators can be a part of a bigger system as in a phase locked loop (PLL), predicting the *start-up* delay remains critical as it determines on how fast the PLL stabilizes, meaning locks to the desired frequency [6]. While the designer can establish constraints that ensure a stable ("good") oscillation, the unpredictable nature of the offset can make the oscillation to look "bad" or "ugly" and hence can make the PLL fail to lock. Hence, given a range of constraints, to show how fast can a circuit settle to oscillation ("start-up" delay) still remains an open research problem.

From a functional verification perspective, the popular verification approaches in the form of assertion/statistical techniques can sometimes exhibit violation that may not be associated with real design failures. For instance, if the output trace of a non-ideal circuit follow the trace of an ideal circuit for say 99.9% and violates for just 0.1% of the simulation time due to false spike in the simulator, then assertion/statistical methods will report a bug in the design. In such cases, the quantification methods fall short to enumerate the method of failure for the circuit behavior appropriately. This report addresses the above issues, by automatically ensuring the correctness of analog circuits in the presence of *offset* conditions using the concept of pattern matching.

The pattern matching techniques are commonly applied to the characterization and validation of high-speed analog and digital circuits. Quite often they are associated with the study of crosstalk, coupling, delays in the data transmission lines [8] during the post-layout and board-level signal integrity (SI) analysis. CAD tools for SI analysis [15] provide a unique waveform comparison capability that can ensure a reliable high-speed data transmission, achieved through interconnect characterization and lab measurements [2]. In the current state-of-the-art, SI analysis can be performed only on the circuit-level simulation traces and board-level design waveforms. In general, any SPICE based simulator can generate SI analysis models for an analog circuit which then could be ported to any standard CAD tool to determine the quality of the simulation traces through waveform comparison. Such a specific trace comparison method can assist the designers to examine the design failures for validity.

To resolve the issue of false violation, the approach based on quantitative methods has to be complemented with a more meaningful analysis of the circuit simulation trace. In the current design/verification flow, the missing qualitative assessment of an AMS design at a higher level of abstraction can be achieved by extending the pattern matching concepts developed in SI analysis to the functional verification. As depicted in Figure 1, the verification based on pattern matching will also help to address the question of *"How to decide on the acceptance/rejection of a circuit simulated with N different process conditions and by N different designers?"*

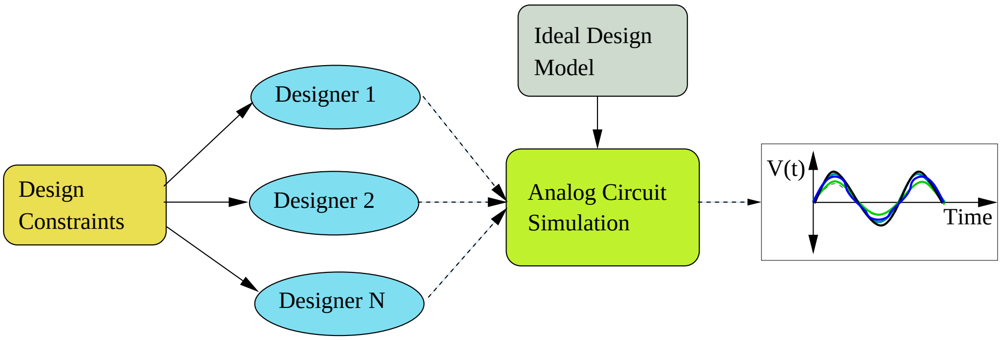


Fig. 1: Analog Verification.

As most of the SI based waveform comparison algorithms are propriety to the tool developers, the challenge is to develop an algorithm that is tailored towards the AMS verification. This report takes a look at a popular pattern matching algorithm that is based on dynamic programming [4]. The underlying idea of this algorithm is to find the subsequence simulation trace between a set of analog signal traces and combine hypothesis testing to determine the probability of failures. Hypothesis testing [12] is the use of statistics to determine the probability that a given hypothesis is true. The statistical property, is expressed as a null hypothesis and in the end, a circuit is *accepted/rejected* with a certain confidence level and error margin.

The Longest Common Subsequence (LCS) is a pattern matching algorithm that finds its applications in computational biology, chip layout design, and so on [17]. In DNA matching, the idea is to find the longest subsequence common to all sequences in a set of sequences [4]. As opposed to the traditional approach of comparing the output of the design to its specification value, we can extend the LCS algorithm to estimate in terms of percentage the exact (100%) or *"closely"* matched simulated output relative to the ideal circuit output. By doing so, instead of blindly rejecting the circuit that violates the specification, designers will have more information during the evaluation and hence can make viable decisions. The main advantage of the pattern matching based approach is that the whole verification process is independent of the circuit models and can be applied to perform a qualitative assessment of any black-box design. The LCSS algorithm that has been presented in [14] cannot handle offset conditions which is addressed in this report.

In analog designs, a *"closely"* matched relation can be defined in many different ways. Let us consider V_1 and V_2 as the output sequences of an ideal and a non-ideal circuit, respectively. First, we say that two output sequences of values V_1 and V_2 are similar if one is a subsequence of the other [4]. Alternatively, another way to measure the similarity between V_1 and V_2 is by finding a third longest sequence of values V_3 that appear in each of the sequences V_1 and V_2 [4]. In reality, it is difficult to find a one-to-one mapping between V_1 and V_2 and hence, designers have to define an acceptable tolerance range for the output as a part of the specification. Thereafter, the LCSS algorithm is defined to quantify the simulated output relative to a specification template.

2 Proposed Methodology

Figure 2 shows the circuit-level simulation methodology that involves parametric and MonteCarlo simulation for a specified technology process. First, we begin with an analog circuit description as a schematic entry that is simulated for a specified process and for a specific initial condition using a SPICE simulator [16]. Parametric analysis involves sweeping multiple parameters to help analyze the stability of a solution within the specified tolerance zone. On the other hand, the basic idea behind the MonteCarlo method is to sample the model of the true population of interest and then to determine the statistical outcome of the simulation. The sampling and calculation procedure is repeated for "M" trials.

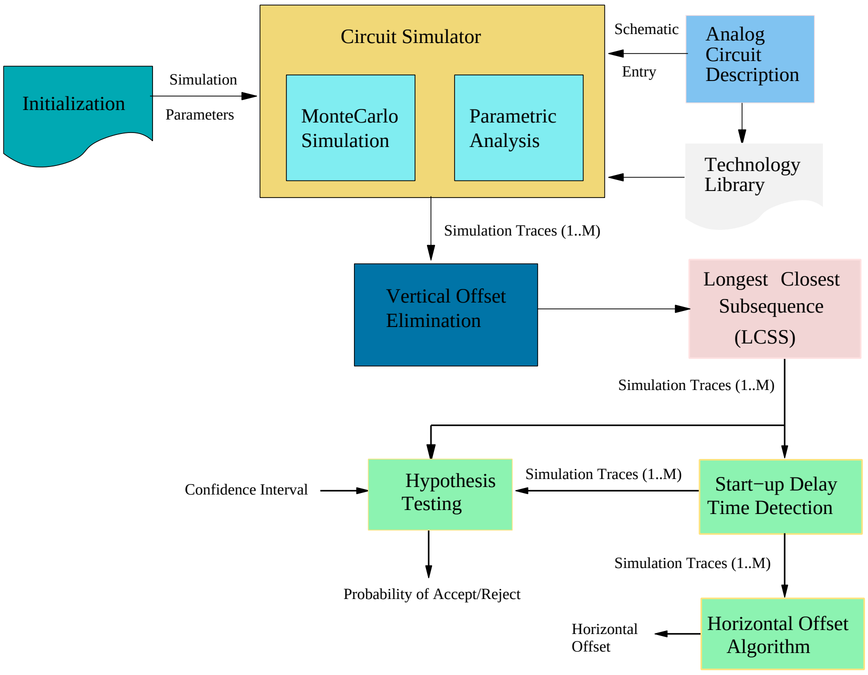


Fig. 2: Circuit Level Simulation

For analog circuits, the MonteCarlo technique is used to study the effect of random variations due to process and mismatch. Mismatch could be either "systematic" where values are fixed and known or it could be "random", where the values are generated randomly and often unknown [16]. In general, there is no theory that governs the number of trials in a MonteCarlo simulation. However, a trade-off exists between the number of trials and the simulation run times. The higher confidence can be gained by choosing a larger number of samples, but at the cost of run-times [12]. The question that has to be answered now is: "*how to decide on the sequences that have offset conditions?*"

As depicted in Figure 2, LCSS and hypothesis testing are combined to process variation and parametric analysis for the circuit level simulation traces that have offset conditions to determine the probability of acceptance/rejection.

2.1 Start-up Delay Time Detection

In general, the start-up delay time can be considered as one kind of horizontal offset. When LCSS algorithm is applied, if the non-ideal trace does not find a match from its first values, it is considered as a start-up delay time as shown in the example in Figure 3.

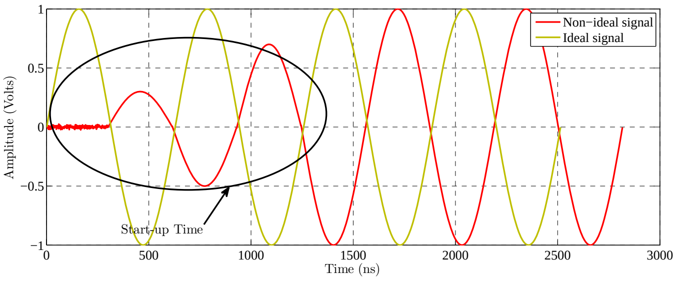


Fig. 3: Start-up Delay Time

Algorithm 1 : Start-up Algorithm

```
Require: $deleted\_values$
$d \leftarrow deleted\_values$
$l \leftarrow length(d)$
for $j \leftarrow$ 2 to $l$ do
    $sp \leftarrow d(j) - d(j-1)$
end for
$spacing \leftarrow eliminate(sp < dist)$
$ls \leftarrow length(spacement)$
$MeanSpacing \leftarrow \sum(spacing)/ls$
$SMS \leftarrow MeanSpacing * SecurityCoffiecient$
$ind \leftarrow 1$
while $d(ind+1) < d(ind) + SMS$ do
    $ind \leftarrow ind + 1$
end while
$index \leftarrow d(ind)$
$startup \leftarrow time(index)$
return $startup$
```

Values in the circled region are out of the interval defined by the tolerance level p $[X - p, X + p]$. We determine the start-up delay time based on the number of deleted points that fall inside the circle and are greater than the number of points in the other regions. For "M" trials, each representing different circuits, rather than using a mathematical description to determine the start-up delay time, it would be more advantageous to determine it directly and automatically from the output traces. The implementation of the start-up delay time detection is given in Algorithm 1.

The algorithm calculates the distance between every two deleted points (lines 3-5). If this distance is greater than a threshold as defined by the user (line 6), then this value will be taken. Otherwise, it represents two successive deleted regions for which the arithmetic mean to estimate the spacing (line 8) distance is done. The user also has to specify a security coefficient usually (< 1) to be multiplied by this spacing distance (line 9). In all cases, it is assumed that the start-up delay time to occur from the time 0s which is very natural (line 10). The spacing distance is then incremented (lines 11-13) to estimate the distance between two deleted points when it is not a start-up time.

2.2 Horizontal Offset

A horizontal offset consists of a shift between two output traces in the time domain as shown in Figure 4. This shift in time is represented as a shift in the index on the two sets of sequences. As LCSS performs operations on a set by set basis rather than the value by value, detecting or eliminating offsets will depend on the correlation between the ideal and non-ideal sequences. The implementation of horizontal offset is described in Algorithm 2.

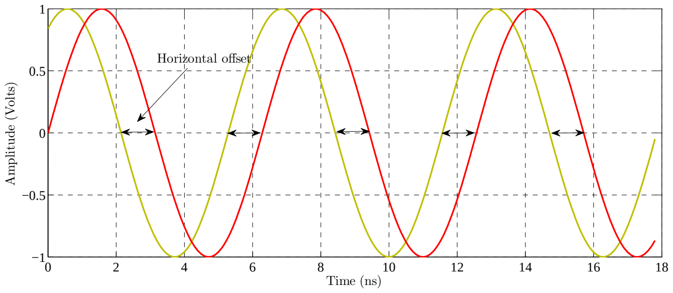


Fig. 4: Horizontal Offset

The horizontal offset can be calculated using the MATLAB built-in correlation function ($xcorr$). The correlation between two signals is maximal when they are aligned. The horizontal offset time (line 3) is thus measured by detecting the index of the maximum of correlation as described in Algorithm 2 (line 2).

Algorithm 2 : Horizontal Offset Algorithm

```
Require: $X$, $Y$,
1: $[cc,\ lags] \leftarrow xcorr(X,\ Y)$
2: $offIndex \leftarrow max(cc)$
3: $offset \leftarrow time(offIndex)$
4: return $offset$
```

2.3 Hypothesis Testing

Hypothesis testing [12] is the use of statistics to make decision about acceptance or rejection of some statements based on the data from a random sample, meaning, to determine the

Table 1: Statistical Estimation Error

	H_0 **is True**	H_1 **is True**
Accept H_0	Correct Decision	Wrong Decision - Type II Error
Reject H_0	Wrong Decision - Type I Error	Correct Decision

probability that a given hypothesis is true. Hypothesis testing in general has two parts:

1. *Null hypothesis*, denoted by H_0, which is what we want to test (e.g., $jitter\_period \leq 3.2\ ns$), and
2. *Alternative hypothesis*, denoted by H_1, which is what we want to test against the null hypothesis (e.g., $jitter\_period > 3.2\ ns$).

If we reject H_0, then the decision to accept H_1 is made. The conclusion is drawn with certain probability of error for a specific confidence interval as summarized in Table 1. The error associated with such statistical estimate for a given confidence interval can be classified to be [10]:

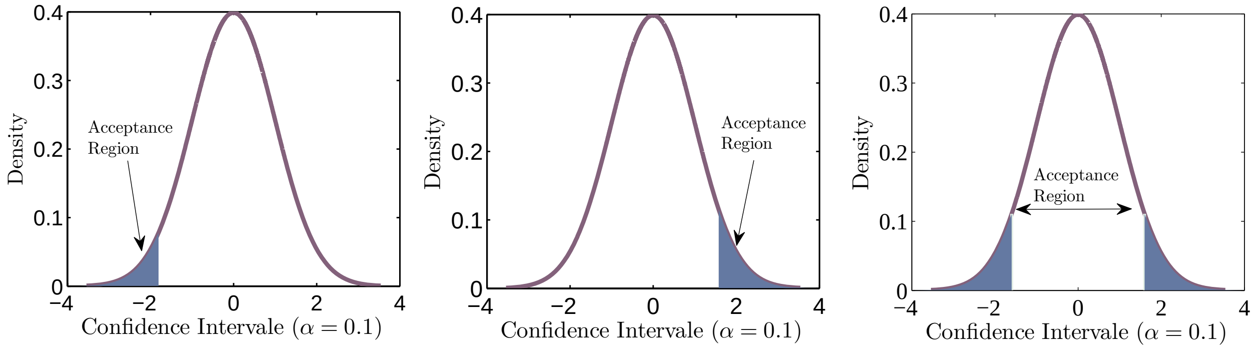
Type I or False positive - H_0 is rejected when it is in fact true.

Type II or False negative - H_0 is true when it is in fact false.

The quantification of error can be made by measuring the probability of accepting/rejecting H_0 when it is actually true/false, respectively. If α and β denote such probabilities then, mathematically they can be represented as

$$\begin{aligned} \alpha &= Pr\{\ reject\ H_0 \mid H_0\ is\ true\ \} \\ \beta &= Pr\{\ accept\ H_0 \mid H_0\ is\ false\ \} \end{aligned} \tag{1}$$

The choice to accept or reject is determined by the direction with which the null hypothesis is proved to be true or false. This direction is decided based on a one-tailed test (*upper* or *lower*) or a *two-tailed* test as shown in Figure 5.



Lower Tail Test

Upper Tail Test

Two Tailed Test

Fig. 5: Accept/Reject Hypothesis Testing

The *upper tail* distribution represents the rejection region for the case where a large value of the test statistic provide evidence for rejecting H_0. On the other hand, a *lower* tail distribution is used if only a small value of the test statistic show proof of H_0 rejection [5].

The *bounded hypothesis testing* [5] also called the *two-tailed* test is determined by a bounded region $[x_1, x_2]$, such that such that H_0 satisfies the following:

$$H_0 : P(x_1 < X < x_2) = P(X < x_2) - P(X < x_1) = 1 - \alpha \tag{2}$$

For instance, $\alpha = 0.05$ and $\alpha = 0.01$ refer to the confidence level of 95% and 99%, respectively. For the case, where the confidence interval is divided equally between the lower and upper bounds, the probability can be determined as follows:

$$\begin{cases} P(X < x_1) & = & \dfrac{\alpha}{2} & = & 0.05 \\ P(X < x_2) & = & 1 - \dfrac{\alpha}{2} & = & 0.95 \end{cases} \tag{3}$$

In any of the above hypothesis testing measures, if the observed sample data over a given interval is within some critical region, then we reject the null hypothesis H_0, else we accept H_0 as shown by the shaded region in Figure 5. In general, the steps in statistical hypothesis testing can be summarized as follows:

1. State the null and alternative hypothesis.
2. Take a random sample from the population of interest.
3. Estimate the statistical measure related to the null hypothesis.
4. Interpret the results to make a decision about acceptance/rejection of the null hypothesis using *critical value* or *p-value* approach with certain standard error.

The *critical-value* approach [5] determines a critical region in which the null hypothesis will be rejected. It depends on the type of tail test (*upper lower* or *two tailed*), observed value and the significant level α. The observed value T_{obs} is calculated based on the sample mean $\bar{x}$, the mean value under the null hypothesis and standard error $\bar{\sigma}$ as described below,

$$T_{obs} = \frac{\bar{x} - \mu_0}{\bar{\sigma}} \tag{4}$$

If the observed value T_{obs} is greater than the critical value, we reject the null hypothesis H_0 otherwise, we retain H_0. The *P-value* approach [5] involves defining the probability of the test statistic to be in the direction of the alternative hypothesis, when the null hypothesis is true. If the derived P-value tends to be smaller it is more likely to reject H_0.

The accuracy of the hypothesis testing depends on how good the sample statistics (*mean*, *variances* and *percentiles*) that determines the standard error are estimated. Sampling by far is concerned with the selection of a subset of the observed data to make a desired statistical inference. Based on the sampling method used one may be able to derive different standard errors and hence the accuracy of the results may vary during hypothesis testing.

For a given analog circuit, every output simulation trace is considered to be a random variable X. For a specified confidence level, a *two-tailed* test can be applied to decide on the acceptance/rejection of the circuit. The detailed procedure for bounded hypothesis testing is illustrated in Algorithm 3.

The first step is to determine the kind of distribution associated with the output simulation trace. It is quite natural to assume a normal distribution for the outputs, however, the variation due to technology and mismatch may sometimes lead to other distributions. Lines (1-19) take into account different distributions and in turn deduce the cumulative distribution function ($CDF(x)$). This is followed by the search for "lower" and "upper" bounds of the critical value that satisfy Equation (3) (lines 20-27). Both the "lower" and "upper" bounds define the acceptance region (where H_0 is accepted) for every random variable X.

Algorithm 3 : Hypothesis Testing (Two-Tailed Test)

```
Require: $Distribution$, $Parameters$
if ($Distribution = LogNormal$) then
    $\sigma \leftarrow Parameters(1)$
    $\mu \leftarrow Parameters(2)$
    $\gamma \leftarrow Parameters(3)$
    $CDF(x) \leftarrow \Phi\left(\frac{\ln(x-\gamma)-\mu}{\sigma}\right)$
else
    if ($Distribution = Normal$) then
        $\sigma \leftarrow Parameters(1)$
        $\mu \leftarrow Parameters(2)$
        $CDF(x) \leftarrow \Phi\left(\frac{x-\mu}{\sigma}\right)$
    else
        if ($Distribution = Weibull$) then
            $\alpha \leftarrow Parameters(1)$
            $\beta \leftarrow Parameters(2)$
            $\gamma \leftarrow Parameters(3)$
            $CDF(x) \leftarrow 1 - \exp\left(-\left(\frac{x-\gamma}{\beta}\right)^{\alpha}\right)$
        end if
    end if
end if
$lower \leftarrow Initial\ Value\ Low$
while $CDF(Lower) \leq 0.05$ do
    $lower \leftarrow lower\ +\ Step$
end while
$upper \leftarrow Initial\ Value\ Up$
while $CDF(Upper) \leq 0.95$ do
    $upper \leftarrow upper\ +\ Step$
end while
return $lower, upper$
```

3 Application - Rambus Ring Oscillator Circuit

Rambus Ring oscillator circuits [9] present a unique problem of lock-up. Unlike the traditional ring oscillator that has odd number of inverters, the Rambus ring oscillator consists of an even number of stages (say *"n"*) , with a bridge (labeled cc) between each stage as shown in Figure 6.

If the "forward" inverters (labeled *fwd*), are much larger than the "cross-coupling" inverters (labeled *cc*), then the circuit acts like a ring of *2n* inverters and will not oscillate. Same problem occurs if the *cc* inverters are much larger than the *fwd* inverters. The oscillation also depends on the circuit initial condition. While designers can establish conditions that ensure a stable ("good") oscillation, offset can sometimes make the oscillation look "bad" or "ugly". The challenge for the verification engineers is to answer the question *"how to judge the quality of oscillation?"* and *"how can we detect the offsets automatically?"*

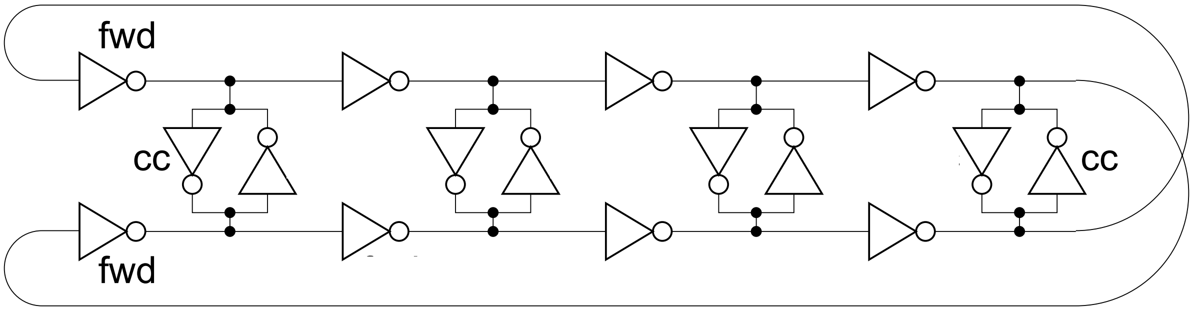


Fig. 6: Rambus Ring Oscillator Circuit.

To better answer the question, the output of the circuit is analyzed by sweeping the transistors size (ratio r) and the initial condition (parametric analysis). Then, MonteCarlo simulation for 100 trials is performed to study the impact of *90nm* technology variation and mismatch on the output behavior of the oscillator. In both cases, LCSS is applied to the output sequence to determine the probability of matching and start-up delay time. In general, transistor sizing ratio is defined as [7]:

$$r = \frac{size\ of\ cc}{size\ of\ fwd} \tag{5}$$

Parametric Analysis:

Based on our simulation and results reported elsewhere [7], it is concluded that the oscillator is unstable (so it oscillates) if $r \in [0.52,\ 2.61]$ with the initial condition equal to the supply voltage for *90nm* technology. Though, the authors in [7] have demonstrated that the circuit will enter an oscillation stage from any initial condition, the challenge would be to verify the quality of oscillation in terms of offsets and start-up delay time. By doing so, the designers will have the leverage to trade-off between the desired frequency range and the speed with which the oscillator can start, meaning the delay time. This is needed as the ring oscillator could be a part of a larger analog mixed signal (AMS) circuit such as PLL, where the start-up delay time can be related to the lock time.

For parametric analysis, the idea is to sweep r for each initial condition and then compare the output simulation trace with an ideal oscillator which have the same frequency using pattern matching algorithm LCSS. The output traces are saved in a **.dat* format and for *p=2%* tolerance, LCSS is applied to determine the percentage matching as summarized in Table 2.

r	Frequency (MHz)	Percentage of matching	Startup Delay Time	Horizontal Offset (ns)
0.52	89.28	71.392	7.3e-9	3.3
0.53	88.88	71.432	4.85e-9	5.15
0.54	89.28	71.682	3.75e-9	3.95
0.55	88.88	71.522	2.75e-9	4.7
0.6	87.71	70.682	5e-11	6.25
0.7	86.20	69.423	5e-11	7.55
0.8	84.03	68.183	5e-11	7
0.9	82.3	67.023	5e-11	7.9
1	80	66.033	5e-11	7.3
1.1	78.12	64.843	5e-11	8.25
1.2	75.47	63.923	5e-11	7.8
1.3	73.26	62.753	5e-11	5.55
1.4	70.92	62.013	5e-11	5.15
1.5	68.25	60.953	5e-11	9.35
1.7	65.35	59.884	5e-11	5.45
1.8	58.99	58.134	5e-11	6.85
1.9	55.55	57.174	5e-11	10.85
2	51.81	56.274	5e-11	11.7
2.1	47.61	55.174	5e-11	12.4
2.2	43.01	54.094	5e-11	9.3
2.3	37.59	52.604	5e-11	15.8
2.4	31.10	50.444	5e-11	12.8
2.5	22.52	45.655	5e-11	26.35
2.6	6.613	32.476	2.55e-9	82.2

Table 2: Parametric Analysis With Initial Value=1v

Finding the LCSS and Start-up Delay Time: Figure 7 shows the percentage of matching as function of the transistor size ratio r. Moreover, when the initial condition is equal to 0.5 volts and with $r \simeq 0.8$, it can be seen that the percentage of matching (dotted line) compared with the other conditions is small. This significant drop is due to long start-up delay time associated with these parametric values as seen in Figure 7.

As seen from the table the parametric analysis has little effect on the results due to horizontal offset. This is true because at any given point, the components values are taken as fixed, which in reality is not the case for analog circuits. Hence, this leads us to the need for a statistical approach based on MonteCarlo simulation to study the impact of those variations on the output voltage.

MonteCarlo Simulation:
For *90nm* technology and based on the number of trials “M”, there are many outputs for the same design. These outputs are the result of varying components values due to the process and mismatch and follow certain probability distribution. The experiment for “M=‘100” trials is conducted on a SUN UltraSPARC-III with 4GB memory.

Finding the LCSS: MonteCarlo simulation shows more variation in their final probability distribution. From the simulation data through statistical analysis, it can be determined that the percentage of matching fits different distributions for different values of r as summarized in Table 3.

For process variation, it is quite natural to find Normal and Lognormal distributions [1] (Figure 8). However, for certain values of r, the data fits Weibull distribution.

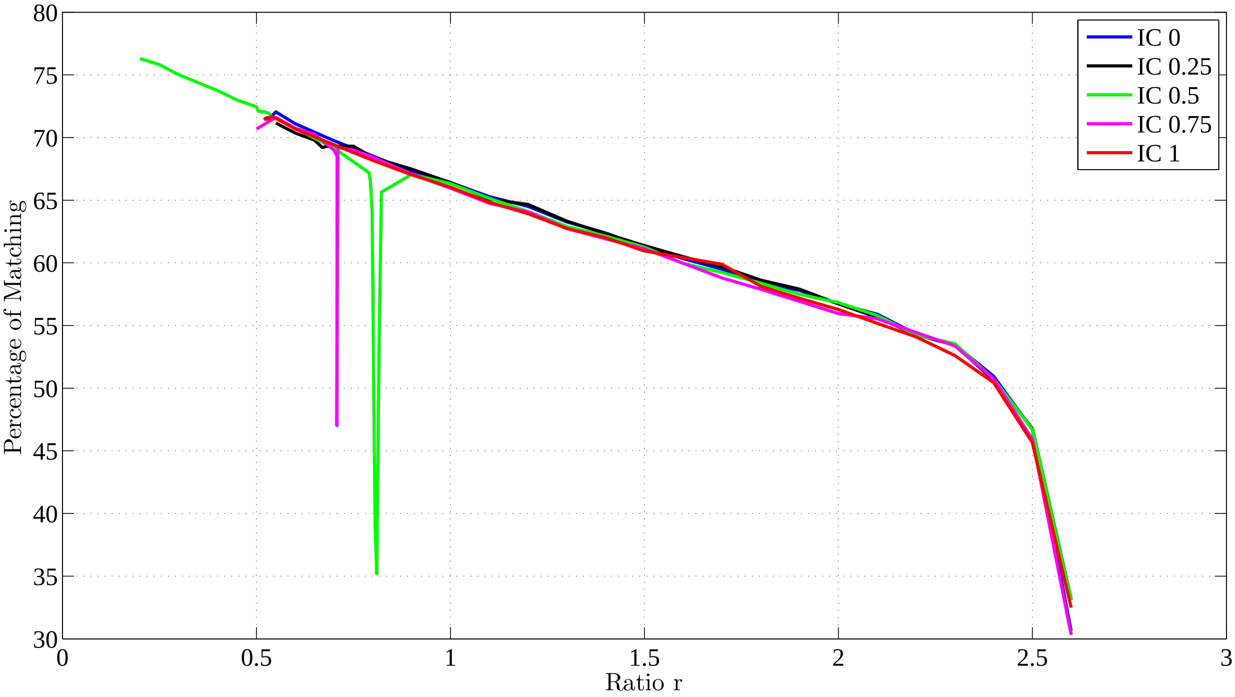


Fig. 7: Percentage of Matching as Function of *r*

Distribution	**r value**
Normal	1.25, 2.25
LogNormal	0.5, 1.75, 2, 2.5
Weibull	0.75, 1, 1.5

Table 3: Distributions of LCSS Percentage Matching

Weibull distribution is a general purpose distribution that can be used to represent normal, exponential and other distributions. For certain values of *r*, the distribution appears to be skewed from its mean and hence for such cases it is better to represent them as Weibull than as Normal and LogNormal distribution. Table 4 summarizes the MonteCarlo results for finding the LCSS and fastest start-up delay time for an initial condition of 0.5 volts.

Start-up Delay Time Estimation: Figure 9 shows the statistical distribution of the MonteCarlo simulation results for estimating the start-up delay time.

Unlike the LCSS percentage matching distribution, the findings are that for r =*1.75*, the curve has different skew with respect to the mean value and does not fit neither a normal nor a Lognormal distribution function. In that case, it falls into Weibull distribution as summarized in Table 5.

From Table 4, it can be noted that the optimal value for the frequency is 82 MHz with 38 out of 100 circuits has maximum percentage matching. When sufficiently large number of trials are used in MonteCarlo Simulations, as per the central limit theorem [12], the estimation of the mean is fairly accurate.

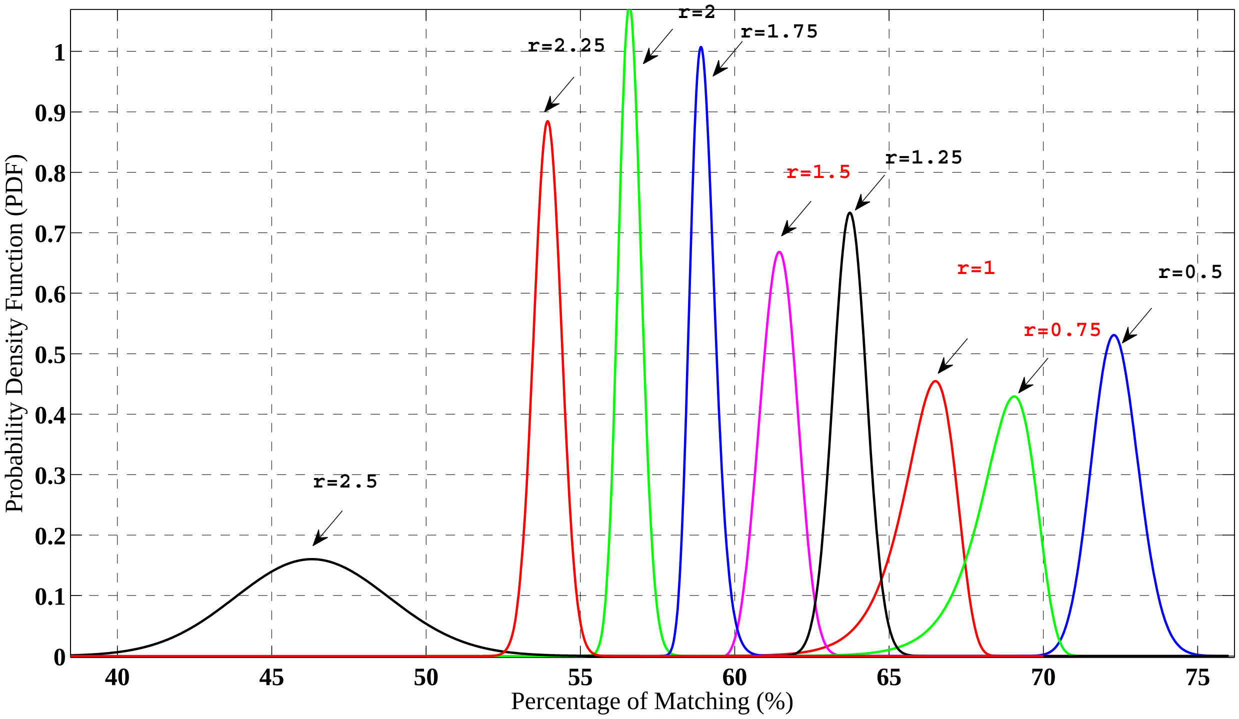


Fig. 8: Percentage of Matching Probabilities

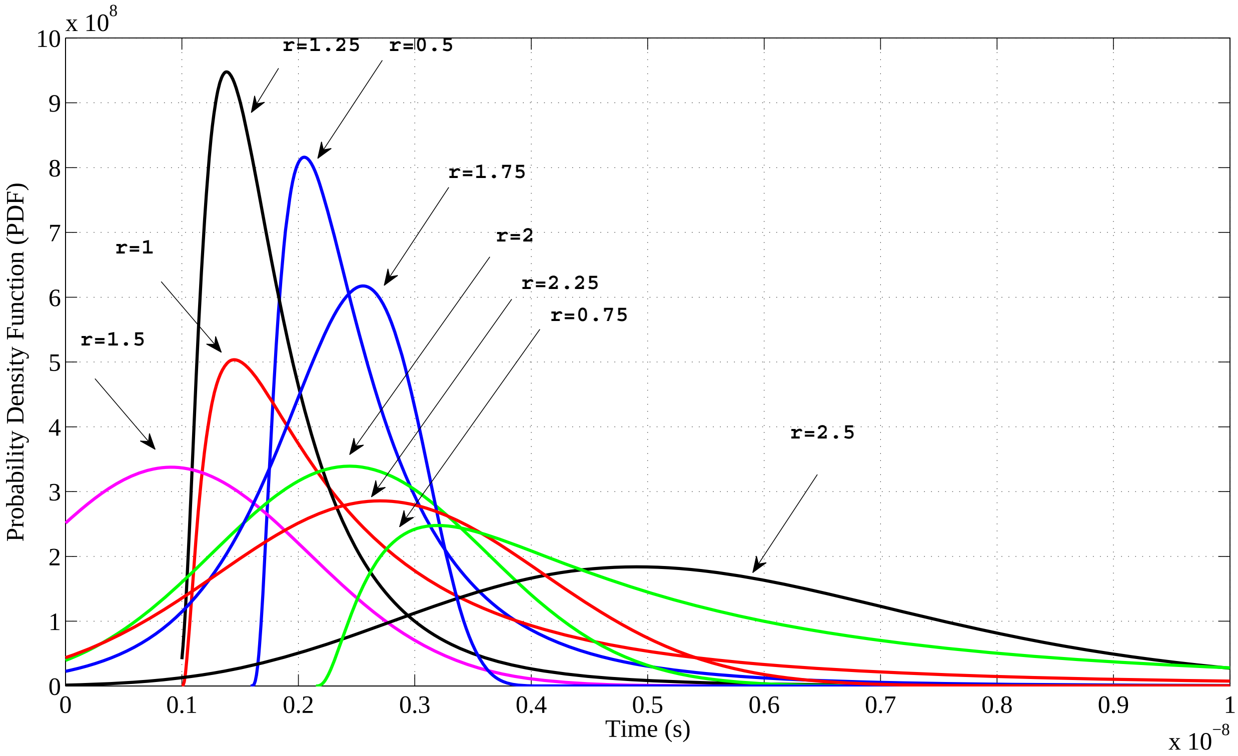


Fig. 9: Start-up Time as Function of r With Monte Carlo Simulation

Decision Based on Hypothesis Testing:

Table 6 summarizes the experiment results for the Rambus Ring Oscillator circuit based

r	**Maximum Frequency (MHz)**	**# of Percentage of Matching[1]**	**Minimum Startup Delay Time (ns)**
0.5	92	19 (72.4%)	2.0
0.75	86	35 (69%)	0.2
1.0	82	38 (67%)	0.1
1.25	76	20 (63.8%)	1.6
1.5	70	19 (61.8%)	40.0
1.75	60	22 (59%)	2.8
2.0	52	26 (56.6%)	3.2
2.25	41	16 (54.2%)	4.0
2.5	24	24 (47%)	6.0

Table 4: Parametric Analysis With Initial Value=0.5volts

Distribution	**r value**
Normal	1.5, 2, 2.25, 2.5
LogNormal	0.5, 0.75, 1, 1.25
Weibull	1.75

Table 5: Distributions of the Start-up Delay Time

on hypothesis testing. It can be seen that, the results for the frequency and percentage matching are consistent with those results (Table 2) found during the parametric analysis, meaning that the mean of the frequency and percentage matching decreases with increase in r. However, compared to standard deviation associated with the frequency, percentage of matching shows different standard deviation but still remains small for *r=2.25*.

r	**Frequency (MHz)**		**Percentage of Matching (%)**		**Start-up Delay Time (ns)**	
	Distribution	**Acceptance region**	**Distribution**	**Acceptance region**	**Distribution**	**Acceptance region**
0.5	LogNormal	[82.11 – 99.28]	LogNormal	[71.13 – 73.61]	LogNormal	[1.82 – 4.65]
0.75	LogNormal	[78.28 – 94.15]	Weibull	[66.51 – 70.01]	LogNormal	[2.69 – 15.05]
1	LogNormal	[73.97 – 88.31]	Weibull	[64.11 – 67.41]	LogNormal	[1.24 – 8.11]
1.25	LogNormal	[68.93 – 81.78]	Normal	[62.84 – 64.63]	LogNormal	[1.16 – 3.37]
1.5	LogNormal	[63.61 – 74.68]	Weibull	[60.48 – 62.37]	Normal	[0 – 2.85]
1.75	LogNormal	[56.67 – 65.95]	LogNormal	[58.38 – 59.71]	Weibull	[0.79 – 3.21]
2	LogNormal	[48.61 – 55.98]	LogNormal	[56.02 – 57.25]	Normal	[0.51 – 4.37]
2.25	LogNormal	[37.88 – 43.93]	Normal	[53.21 – 54.68]	Normal	[0.408 – 5]
2.5	LogNormal	[16.51 – 28.48]	LogNormal	[42.22 – 50.41]	Normal	[0.86 – 10.38]

Table 6: Hypothesis Testing Results

The results derived for the start-up delay time shows the adverse influence of process variation and mismatch. The start-up time exhibits a larger acceptance region where $r \simeq 0.75$ which confirms with the results found in the parametric analysis for the initial condition 0.5v. For $r \simeq 0.80847$ and for the initial condition 0.5v, the oscillator takes a huge time to start in compared to other r values. In summary, the hypothesis test results can be different for different confidence intervals and the accuracy would be compromised if the confidence level is too high or too low. Higher confidence level would increase the error margin and degrade the reliability; lower confidence level on the other hand would increase the rejection region and cause low accuracy.

4 Conclusion

This report describes a methodology based on pattern matching to account for process variation and mismatch in analog circuits. The longest closest subsequence (LCSS) is used at circuit level descriptions for the qualitative analysis on the simulation traces that have *offset* (vertical, horizontal) conditions. The effect of mismatch and process variation at circuit level is studied by performing parametric and statistical analysis to estimate in terms of percentage the closest simulation trace that matches with the simulation trace of an ideal circuit. The efficiency of our approach is illustrated on a Rambus Ring Oscillator circuits for a *90nm* fabrication process.

The conventional verification method may require major changes to the test-bench structure during scaling of analog designs, and still cannot answer the question: *"How do we choose the test set?"* or *"Can we retain the same test points?"* This is because, the test points are chosen in such a way that it represents the limit of operation of the design which of course may or may not change when the designs are scaled. However, LCSS based techniques work on the simulation trace in polynomial time and hence it will be well suited for verifying "black-box" analog designs.

Other future plan is to develop techniques that could handle *frequency* offset conditions and address the issue related to stability of analog circuits. Additionally, the algorithm has to be optimized for speed and memory utilization. For instance, this can be done by using threading techniques for LCSS implementation.

References

[1] B. Ankele,W. Hölzl, and P. O'Leary. Enhanced MOS parameter extraction and SPICE modeling for mixed signal analogue and Digital Circuit Simulation. IEEE International Conference on Microelectronic Test Structures, pp. 133137, 1989.

[2] H. B. Bakoglu. Circuits Interconnects, and Packaging for VLSI. Addison Wesley, 1990.

[3] H. Chang and K. Kundert. Verification of Complex Analog and RF IC Designs. Proc. of the IEEE, 95(3), pp. 622639, 2007.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms, The MIT Press, 2001.

[5] J. E. Freund. Modern Elementary Statistics. Prentice hall, 1984.

[6] P. A. Gray, P. J. Hurst, S. H. Lewis and R. G. Meyer. Analysis and Design of Analog Integrator Circuits. Wiley, 2009.

[7] M. Greenstreet, and S. Yang. Verifying Start-Up Conditions for a Ring Oscillator, ACM Great Lakes Symposium on VLSI, pp. 201-206, 2008.

[8] R. J. Haller. The Nuts and Bolts of Signal-Integrity Analysis. Electronics Design, Strategies, News (EDN), 2000.

[9] K. D. Jones, J. Kim, and V. Konrad. Some "Real World" Problems in the Analog and Mixed Signal Domains. International Workshop on Designing Correct Circuits, pp. 51-59, 2008.

[10] E. L. Lehmann, J. P. Romano. Testing Statistical Hypotheses, Springer. 2005.

[11] L. L. Lewyn, T. Ytterdal, C. Wulff, and K. Martin. Analog Circuit Design in Nanoscale CMOS Technologies. Proc. of the IEEE, 95(10):1687-1714, October 2009.

[12] W. L. Martinez and A. R. Martinez. Computational Statistics Handbook with MATLAB. Chapman & Hall/CRC, 2002.

[13] F. Pcheux, C. Lallement, A. Vachoux. VHDL-AMS and Verilog-AMS as Alternative Hardware Description Languages for Efficient Modeling of Multidiscipline Systems. IEEE Transactions on CAD, 24(2): pp.204-225, 2005.

[14] R. Narayanan, M. Zaki, and S. Tahar: Ensuring Correctness of Analog Circuits in the Presence of Noise and Process Variation Using Pattern Matching, IEEE/ACM Design Automation and Test in Europe, pp. 1188-1191, 2011.

[15] SimDE^{TM} Waveform. http://www.iometh.com/Product/SignalMeth/index.html, 2012.

[16] Synopsys HSPICE User Guide: RF Analysis. http://www.synopsys.com, 2009.

[17] M. Yoshikawa and H. Terai. Constraint-Driven Floorplanning based on Genetic Algorithm. ACM International Conference on Computer Engineering and Applications, pp.147-151, 2007

[18] J. F. Witte, K .A. A. Makinwa and J. H. Huijsing: Dynamic Offset Compensated CMOS Amplifiers. Analog Circuits and Signal Processing, Springer, 2009.