Statistical Runtime Verification of Analog and Mixed Signal Designs

Zhiwei Wang[1], Mohamed H. Zaki[2], and Sofiène Tahar[1]
[1]Dept. ECE, Concordia University, Montreal, Quebec, Canada
Email: {zhiw_wan, tahar}@ece.concordia.ca
[2]Dept. CS, University of British Columbia, Vancouver, British Columbia, Canada
Email: mzaki@cs.ubc.ca

***Abstract*—One of the challenges for the verification of analog and mixed signal (AMS) designs is the stochastic behavior associated. In this paper, we propose a runtime verification approach to verify the statistical property of the AMS design. The methodology is based on the combination of the statistical method and Mont Carlo simulation. The verification procedure produces confidence level and error margin that provide the tolerance and accuracy for the verification results. We apply the proposed methodology to study the jitter property of a PLL design.**

I. Introduction

With the constant growth in integrated circuit technology, more complex functionalities can be realized in compact systems such as smart cell phones and portable game consoles. System on Chip (SoC) architecture has prevailed for the last decade. It usually contains digital, analog, mixed signal and radio frequency functional units in one chip. Although SoC designs have been driving the semiconductor industry, the growth of design productivity has been lagging behind the improvement in the number of transistors per chip by as much as 37% [16]. Analog and mixed signal (AMS) components, which connect the analog world and digital world, are considered a bottleneck in improving the overall performance of the system and the factor for enhancement of the first silicon success rate. The failure of AMS circuits has been one of the major causes for the high design re-spun rate [3]. To overcome these obstacles, 70% of the total design effort in the semiconductor industry is now spent on verification. The design and verification of AMS systems became very important in recent years.

The major challenges for the verification of AMS designs can be regarded as the complexity, the formal specification of the property, the computational consumption and the stochastic behavior. The complexity mainly refers to sophisticated behaviors involved in the AMS design which contains pure analog components, pure digital blocks and the components mixed of the two. The continuous (analog) signal and discrete (digital) signal should be described using the uniform approach in order to exhibits the interrelationship between the two. The lack of formal specification language for AMS properties challenges the verification engineers all the time. Even now, there is no standard language for specifying the property in AMS systems. Due to the complex and mixed behaviors associated within the AMS design, high simulation time as well as expensive memory usage makes the verification of AMS design more challenging. The stochastic behavior results from the undesired variation or the random noise within the AMS design. Unlike other challenges, stochastic behaviors are even responsible for the failure of the AMS circuit. In this paper, we tackle this challenge for the AMS design.

Runtime verification is an approach, which bridges the gap between traditional simulation and formal verification, to verify the property at runtime. Runtime verification deals with the detection of violation, as well as satisfaction, of the property. A monitor is used to detect the violation. The monitoring technique can be performed in two ways, namely, online and offline monitoring. Online monitoring, which is used to check a current execution of a system when the simulation is running, is able to detect a property violation as soon as it occurs. On the other hand, offline monitoring operates on a set of recorded executions after the simulation is done. When the reliable issue, such as random behaviors, is considered, we prefer offline monitoring.

In this paper, we propose a runtime verification methodology for statistical properties of AMS designs in an offline fashion. The approach combines the statistical method and Monte Carlo simulation to investigate the stochastic behavior by verifying the statistical property of the AMS design. Confidence level and error margin are provided along with the verification results.

Related work Statistical verification can be divided into three main categories: statistical theorem proving, statistical model checking and statistical runtime verification. Although several interesting advances have been made in statistical theorem proving, this technique is still in its infancy. The theorem for continuous random variables and random processes is needed to handle the analysis and verification of AMS and hybrid system [4]. The model checking method has been advanced first to complement general model checking. In [14], the authors present an independent model checking approach for verifying probabilistic properties of discrete event systems. The probabilistic properties were expressed using continuous stochastic logic (CSL) [1] formulas. These formulas were then verified through Monte Carlo simulations and statistical hypothesis testing. The verification procedure provides two parameters, α and β, which represent the probability of making a wrong decision in checking whether a formula is true or false. In a related work [15], the author presents a probabilistic model checking method to bound the probability of error, mentioned in [14], for the indifferent region (i.e., the region where both acceptance and rejection decisions can not be made). A symmetric polling system was studied to demonstrate the performance of the method. Following the statistical model checking approach in [14], [15], the authors in [2] applied this technique to a class of AMS circuits for the first time. The saturation property of a third order delta-sigma converter was verified both in time and frequency domains. However, the issues of state explosion and excessive computation time still prevail in statistical model checking. In this paper, we employ the statistical runtime verification to avoid this issue.

Statistical runtime verification has also been investigated in the past. One of the interesting works is [13] where the authors introduce a methodology to verify quantitative and probabilistic properties in a real-time system at runtime. The quantitative specification was realized using Meta Event Definition Logic (MEDL)which is based on LTL. Statistical hypothesis testing technique was employed to evaluate the probabilistic properties and to make decisions about acceptance and rejection. Whenever the decision is made, a confidence level and error margin is provided. The monitor was implemented in a runtime verification tool termed MaC (Monitoring and Checking) [5] and performed in an online fashion. Instead, in this paper, we present a methodology for statistical runtime verification for AMS designs and the impact of confidence level for the verification results is discussed. The monitor is implemented using Monte Carlo hypothesis testing. We provide the information of error margin in addition to the confidence level for the verification results.

The rest of the paper is organized as follows: in Section II we introduce the essentials of the statistical hypothesis testing and Monte Carlo simulation. The statistical runtime verification methodology is described in detail in Section III. The experimental results are presented in Section IV. Finally, in Section V we conclude the paper.

II. Preliminaries

A. Statistical Hypothesis Testing

Statistical hypothesis testing is a technique which provides a decision making procedure about logic statements based on statistical information. The conclusion is drawn with a confidence level and an error estimate. Hypothesis testing is generally formulated in two parts. They are *null hypothesis*, denoted by H_0, which is what we want to test and *alternative hypothesis*, denoted by H_1, which is what we want to test against the null hypothesis. If we reject H_0 based on our statistical investigation, then the decision to accept H_1 is made.

Error Bounds

There are two kinds of error bounds that apply when we are making a decision in statistical hypothesis testing. They are known as Type I error and Type II error [9]. A Type I error, or false positive, occurs when we reject H_0 which is actually true. A Type II error, or false negative, arises when we accept H_0 which is actually false. α and β denote the probability of Type I error and Type II error respectively. Formally,

$$\alpha = Pr\{\texttt{reject } H_0 \,|\, H_0 \texttt{ is true }\}$$

$$\beta = Pr\{\texttt{accept } H_0 \,|\, H_0 \texttt{ is false}\}$$

Confidence Level

In hypothesis testing, the confidence is drawn according to the compliment of the Type I error α. α is also called *significance level*. Formally, the confidence level δ is give by:

$$\delta = 1 - \alpha \tag{1}$$

For instance, $\alpha = 0.05$ and $\alpha = 0.01$ refer to the confidence level of 95% and 99%, respectively. Before performing the hypothesis testing, the Type I error should be established.

Tail Test

The *rejection region* is needed to perform the statistical hypothesis test. A rejection region, over which we would reject H_0, is the area covered by the probability density function (PDF). The *critical value* is used to divide the domain of the test statistic into a rejection region and a non-rejection region. Generally, the rejection region is located at the tails of the distribution of the test statistic when H_0 is true. The test can take place either in the lower tail or the upper tail which depends on the alternative hypothesis H_1. *Upper tail test*: If a large value of the test statistic would provide evidence for rejecting H_0, then the rejection region is in the upper tail of the distribution of the test statistic. *Lower tail test*: If a small value of the test statistic would provide evidence for rejecting H_0, then the rejection region is in the lower tail of the distribution of the test statistic.

B. Monte Carlo Simulation

Monte Carlo method originated in the 1940's [11]. It refers to a method of solving problems using random variables. It is widely used in the estimation of phenomena involving stochastic processes. The basic idea behind the Monte Carlo method is to sample the model of the true population of interest. This is followed by calculating the statistics of interest. The sampling and calculation procedure is repeated for M trials. The investigation of the distribution characteristics of the statistics is carried out based on those M experiments. When the Monte Carlo method is applied in hypothesis testing, we sample from a distribution which is known or assumed. The Monte Carlo hypothesis testing algorithm used for statistical runtime verification is described next.

III. Methodology

The statistical runtime verification is carried out using Monte Carlo monitor to check the statistical property of the AMS design. The monitor is constructed by hypothesis testing and Monte Carlo simulation. In this section, we first present the overall methodology of the statistical runtime verification. Then, we describe how to apply Monte Carlo simulation to the statistical hypothesis testing.

A. Statistical Runtime Verification

The methodology of the statistical runtime verification is shown in Figure 1. The statistical property, such as mean or variance, we want to verify is expressed as a null hypothesis H_0. The alternative hypothesis H_1 becomes the counterexample naturally. The Monte Carlo monitoring is then carried out based on the confidence level δ we specify. The decision is made based on the significance level α with respect to the confidence level δ. In Monte Carlo Monitor, the property is verified using hypothesis test incorporated with Monte Carlo simulation. The statistical property is verified if the decision of accepting the null hypothesis H_0 is made. The rejection of null hypothesis H_0 leads to the violation of the property. All the decisions are produced under the specific confidence level δ along with the error margin ϵ. The error margin specifies a confidence interval where the estimated statistic falls with the probability of δ. The Monte Carlo simulation is then employed to evaluate the performance of the hypothesis test in terms of the Type I error. The difference between the significance level α and the actual Type I error $\hat{\alpha}$ committed during the procedure provides a performance indicator for the hypothesis testing.

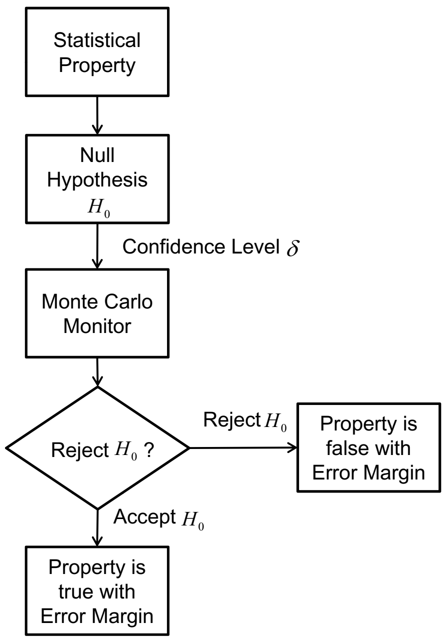

Fig. 1: Monte Carlo Based Statistical Runtime Verification

B. Hypothesis Testing - Critical Value Approach

There are several approaches for hypothesis testing. In this paper, we address only the critical value approach. It is also important to note that in order to perform the hypothesis test, the distribution of the test statistic under the null hypothesis H_0 is assumed to be known. Before we describe the critical value approach, we first introduce an important concept, namely quantile function.

Quantile Function

Quantile function plays an important role in statistics [9]. The quantile q_p of a random variable X is defined as the smallest number q such that the cumulative distribution function (CDF) $P(X)$ is greater than or equal to some p, where $0 < p < 1$. This can be calculated for a continuous random variable with probability density function $f(x)$ by solving

$$p = \int_{-\infty}^{q_p} f(x)dx \tag{2}$$

for q_p. The quantile function is the inverse of the cumulative distribution function (CDF) and is given by

$$q_p = quantile(p) = F^{-1}(p) \tag{3}$$

The p-th quantile of a random variable X is the value q_p such that

$$F(q_p) = P(X < q_p) = p \tag{4}$$

In hypothesis test, we will see that quantile function is used to determine the decision about rejection of a hypothesis.

Critical Value Approach

In hypothesis testing, if the observed statistic is within some region, we reject the null hypothesis. The interval where the null hypothesis is rejected is called *critical region*, or *rejection region*. The *critical value* is used to divide the domain of the test statistic into rejection region and non-rejection region. The critical value approach is used to check whether the observed value falls into the rejection region. It requires α and T_{obs} to calculate the *critical value*. α is the significant level, or Type I error, and T_{obs} is the observed value calculated by

$$T_{obs} = \frac{\bar{x} - \mu_0}{\bar{\sigma}} \tag{5}$$

where $\bar{x}$ is the sample mean of the random variable, μ_0 is the mean value under the null hypothesis and $\bar{\sigma} = \sigma_x/\sqrt{n}$ is the standard error of the sample.

The probability of wrongly rejecting H_0, or Type I error, is supposed to be controlled before we perform a hypothesis test. The critical value depends on the significance level α, namely the Type I error. The typical values of α are $0.01, 0.05$, and 0.10. The critical value is found as a quantile (under the null hypothesis H_0) calculated using Equation 3.

C. Monte Carlo Hypothesis Testing

In order to perform Monte Carlo monitoring, the distribution of the population is supposed to be known in advance or assumed. Then a model which reflects the characteristics of the original population is made. Monte Carlo simulation is used to generate random sample for estimation of the distribution of the original population. Hypothesis testing is then used to check whether observed value of test statistic falls into the reject region specified by the estimated critical value.

The detailed procedure is illustrated in Algorithm 1 where T_{obs} is observed value calculated using Equation 5, n is the sample size, σ is the population standard deviation, $\bar{\sigma} = \sigma/\sqrt{n}$ is the standard error of the sample, and μ denotes the population mean. The loop between line 1 and line 5 is the

Monte Carlo simulation repeated for M trials. In each trial, we randomly sample from the distribution of population under the null hypothesis with the same sample size n (line 2 and line 3) and then calculate and record the observed value of this pseudo sample T_{mc} (line 4) which is given by

$$T_{mc} = \frac{\bar{s} - \mu}{\bar{\sigma}} \tag{6}$$

where $\bar{s}$ is the mean value of the pseudo random sample. It is important to note that all the calculations till now are under the hypothesis that H_0 is true. The hypothesis testing is performed afterwards (from line 6 to line 21). In the case of the lower tail test the significant value is α, while $1 - \alpha$ is chosen when we perform the upper tail test. For upper tail test, the large significant evidence is investigated. In other words, if the observed value T_{obs} is greater than the critical value we reject the null hypothesis H_0. Otherwise, we retain H_0. For lower tail test, a small value is needed as the evidence to reject H_0. Hence, if T_{obs} is less than the critical value, calculated using α in this case, we reject H_0. Otherwise we retain H_0. In Monte Carlo hypothesis testing, the critical value is estimated based on the model generated by Monte Carlo simulation. Whereas the critical value in Algorithm 1 is based on the standard normal distribution. The hypothesis test is carried out using the estimated critical value and the observed value T_{obs} the same way that Algorithm 1 does. Because for each hypothesis test the Monte Carlo simulation generates different random model, it is expected that the estimated critical value varies for each test.

Algorithm 1 Monte Carlo Hypothesis Testing

```
Require: $\alpha$, $T_{obs}$, $n$, $\sigma$, $\bar{\sigma}$, $\mu$
 1: for $i = 1$ to $M$ do do
 2:     $r = random\_number\_generator(n)$
 3:     $s = \sigma \cdot r + \mu$
 4:     $T_{mc}(i) = (mean(s) - \mu)/\bar{\sigma}$
 5: end for
 6: while $Upper\ Tail\ Test$ do
 7:     $critical\_value = quantile(T_{mc}, 1 - \alpha)$
 8:     if $T_{obs} > critical\_value$ then
 9:         $Reject\ H_0$
10:     else if $T_{obs} < critical\_value$ then
11:         $Accept\ H_0$
12:     end if
13: end while
14: while $Lower\ Tail\ Test$ do
15:     $critical\_value = quantile(T_{mc}, \alpha)$
16:     if $T_{obs} < critical\_value$ then
17:         $Reject\ H_0$
18:     else if $T_{obs} > critical\_value$ then
19:         $Accept\ H_0$
20:     end if
21: end while
```

D. Error Margin

For random variables, it is unlikely that the observed value of the sample is exactly equal the true value of the population parameter such as the mean or the variance. Hence, it is more useful to have an interval of numbers that contains the true value. The probability of the true value appearing in the interval is the confidence level δ we introduced previously. We define the error margin ϵ as

$$\epsilon = q_{(1-\alpha/2)} \frac{\sigma}{\sqrt{N}} \tag{7}$$

where $q_{(1-\alpha/2)}$ is the quantile for the probability $1-\alpha/2$. The error margin depends on the significant level α and standard deviation σ of the population.

The error margin is provided together with the confidence level when the hypothesis test is done. The confidence level indicates the Type I error, which we establish in advance, of the statistical hypothesis testing. The error margin provides the confidence interval that contains the population parameter we want to estimate. The larger the confidence interval which incloses the population parameter is, the higher confidence level we can achieve.

IV. Experimental Results

Jitter is simply the deviation in time between a noisy signal and an ideal one. It affects the quality of the system especially for high frequencies. A typical PLL based frequency synthesizer, shown in Figure 2, is widely used in communication systems as clock generator or clock recovery circuits.One major source of jitter is the reference clock input. The voltage controlled oscillator (VCO) within the PLL is another major source of jitter noise which exhibits random jitter. In this paper, we focus only on the random jitter associated within the VCO. The PLL system was implemented in Matlab.

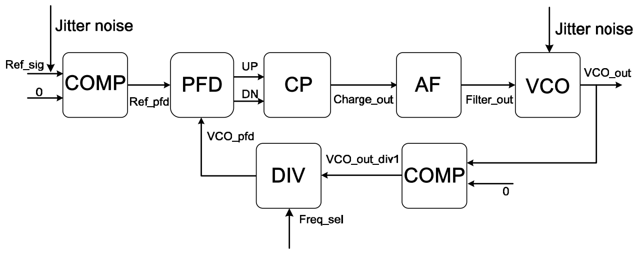

Fig. 2: PLL Frequency Synthesizer with Jitter Sources

A. Jitter Noise in VCO

VCO oscillates with the frequency proportional to the input voltage signal coming from the lowpass filter. The jitter in VCO is mainly caused by thermal noise of the circuit. Hence, it exhibits Gaussian random process [8]. The model of a VCO with jitter noise is illustrated in Figure 3. In fact, VCO generates the sine wave by dealing with the frequency. Hence, the jitter, which is defined as variation in the period, has to be modeled as a variation in the frequency of the VCO. Assume that, the frequency of a periodic signal without jitter is given by $f = \frac{1}{T}$ where T is the period of the ideal signal. The jittery frequency can be represented as

$$f_{jitter} = \frac{1}{T + \Delta T} = \frac{1}{\dfrac{1}{f} + \Delta T} = \frac{f}{1 + \Delta T \cdot f} \quad (8)$$

with $\Delta T = J\lambda$ and J is the jitter deviation and λ is a zero mean unit-variance Gaussian random process. We finally obtain the formula of VCO with jitter noise as

$$VCO_{out}(t) = Acos(\omega_0 t + \phi(t) + \phi_0) \quad (9)$$

$$\phi(t) = K_{VCO} \int_0^t \frac{Filter\_out(\tau)}{1 + \dfrac{J\lambda \cdot K_{VCO} \cdot Filter\_out(\tau)}{2\pi}} d\tau \quad (10)$$

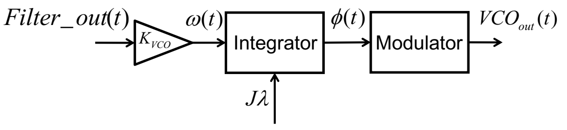

Fig. 3: VCO Model with Jitter Noise

B. Statistical Runtime Verification

We apply the proposed methodology to verify the property of jitter noise in VCO mentioned in the previous section. The property is expressed as: the period jitter of the given system is less than a specific value. The specific value comes from the system specification of phase noise. For example, if the phase noise is $\mathcal{L} = -25$dBc at the offset frequency 10Hz, the corresponding period jitter to this phase noise is calculated as 5.62ns [6]. As a result, the null hypothesis H_0 and the alternative hypothesis H_1 of this property can be expressed as

$$H_0 : J_{period} \leq 5.62ns; \quad (11)$$

$$H_1 : J_{period} > 5.62ns. \quad (12)$$

where J_{period} is the period jitter of the VCO output. We estimate J_{period} by observing the information along the simulation path (0.005s) as a sample. The observed period jitter is denoted by J_{obs}. Since a large value would provide the evidence for the rejection of the null hypothesis H_0, an upper tail test scenario is considered in this case.

The Mont Carlo monitor is implemented using Matlab [10]. The experiments were run on the ULTRA SPARC-IIIi server (177 MHz CPU, 1GB memory). The experimental results for several jitter deviation factors J with the confidence level $\delta = 0.95$ ($\alpha = 0.05$) are shown in Table I. The simulation was carried out under the significance level $\alpha = 0.05$. J varies from 10^{-8}s to 10^{-7}s. The acceptance of the null hypothesis H_0 indicates that the property is satisfied and the rejection of H_0 indicates that the property is violated and the period jitter in VCO is larger than the specification. Due to the upper tail test, the evidence of rejecting H_0 is that the observed value T_{obs} is greater than the critical value based on the significance level. When $J = 5 \times 10^{-7}$s, the Monte Carlo monitor announces the rejection of H_0 based on the fact that T_{obs} is greater than the critical value (i.e., it falls into the rejection region). The experiment was performed with the Monte Carlo trials $M = 1000$. The last column lists the error margins (ϵ) for the confidence interval of 95% when J varies. Each error margin forms a confidence interval with the observed value J_{obs} for J_{period}. For example, when $J = 5 \times 10^{-7}$s, the probability that the true value of the period jitter of the whole simulation path presents within the interval (90.4403, 92.5857)ns is 95%. The interval falls into the rejection region which indicates that we have the confidence level of 95% to reject H_0 in this case. The error margin ϵ is calculated using Equation 7. It is noted from

TABLE I: Statistical Runtime Verification with Different J

J(s)	$C.V.$	T_{obs}	J_{obs} (ns)	H_0	α	ϵ (ns)
1×10^{-8}	1.5069	-3.8455	3.4609	Accept	0.05	1.0710
5×10^{-8}	1.5896	-1.7013	4.6667	Accept	0.05	1.0589
1×10^{-7}	1.6597	0.5900	5.9552	Accept	0.05	1.0711
5×10^{-7}	1.5442	152.73	91.513	Reject	0.05	1.0727

Table I that when J increases from 1×10^{-7}s to 5×10^{-7}s, the Monte Carlo monitor experiences a procedure that the decision changes from acceptance to rejection. The Table II shows the verification results influenced by the variation of J and α. J increases by a small step from 1×10^{-7}s to 1.4×10^{-7}s. The decision tends to change from acceptance to rejection. However, for certain selection of J, if we change the significance level α, the decision can be different. For example, in the case of $J = 1.2 \times 10^{-7}$s, we accept H_0 when α is 0.05; while we have to reject H_0 when α is 0.1. Figures 4 (a) and (b) show the observed value T_{obs} (small triangle) and the rejection region (shaded area) in the case that $J = 1.2 \times 10^{-7}$s and α is 0.05 and 0.1, respectively. In Figure 4 (a), the observed value is located outside the rejection region. In Figure 4 (b), the rejection region is enlarged and includes the observed value T_{obs}. It can be explained using the critical value approach: the fact that reducing the confidence level δ, or increasing the significance level α, makes the critical value smaller. As a result, the rejection region is enlarged accordingly. If the observed value happens to fall within the enlarged rejection region, the null hypothesis is rejected. Similar situation occurs when $J = 1.3 \times 10^{-7}$s except that when the confidence level increases from 95% to 99% (α from 0.05 to 0.01), the Monte Carlo monitor accepts H_0 instead of rejecting it. This situation is shown in Figures 4 (c) and (d). In addition, we notice that the error margin ϵ increases as the confidence level decreases and vice versa. The reason is that in order to achieve higher confidence level, the interval is supposed to be larger to allow the estimated value to be included there. In other words, the probability that the estimated value falls into the narrower interval is smaller than that for the wider one.

C. Discussion

The hypothesis test results can be different for different confidence levels when the observed value is approaching the

TABLE II: Statistical Runtime Verification with Different J and α

J(s)	$C.V.$	T_{obs}	J_{obs} (ns)	H_0	α	ϵ (ns)
1×10^{-7}	1.6597	0.5900	5.9552	Accept	0.05	1.0711
1.1×10^{-7}	1.6383	0.9244	6.1432	Accept	0.05	1.0730
1.2×10^{-7}	1.6571	1.5768	6.5267	Accept	0.05	1.0953
1.2×10^{-7}	1.1920	1.5768	6.5267	Reject	0.1	0.9052
1.3×10^{-7}	1.5874	1.9480	6.8476	Reject	0.05	1.1482
1.3×10^{-7}	2.4287	1.9480	6.8476	Accept	0.01	1.3280
1.4×10^{-7}	1.6803	2.8203	7.2094	Reject	0.05	1.1057
1.4×10^{-7}	1.3506	2.8203	7.2094	Reject	0.01	1.4374

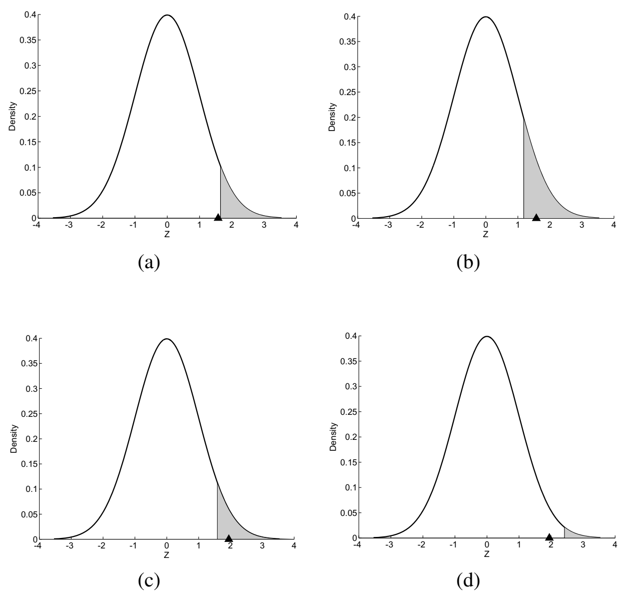

Fig. 4: Effects of Confidence Level Selection

critical value. The accuracy would be affected if the confidence level is too high or too low. On the other hand, the confidence level influences the error margin. 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. The reason is that the interval needs to be enlarged in order to include the estimated value for higher probability, or higher confidence level. 95% of confidence level, which compromises the two situations, is the most commonly used and suitable for most engineering and science researches [12]. There is a dilemma in the proposed statistical runtime verification. Once we intend to increase the confidence level of the test, the accuracy and reliability are compromised. It is not unusually the case when statistical methods are applied. In fact, we estimate the statistical property of the entire population by observing a sample sequence of it. The confidence level of 100% is impossible to reach. There are some techniques to provide more reliable decision with certain confidence level such as to guarantee the confidence interval within the rejection region or non-rejection region.

V. Conclusion

We used the combination of hypothesis testing and Monte Carlo simulation for statistical runtime verification of the AMS design. The hypothesis test makes the decision between the null hypothesis and its exclusive alternative hypothesis. Monte Carlo simulation is used to generate the estimate random model in the case that the distribution of the population is not known. This makes the hypothesis robust to most stochastic processes. We present random jitter analysis using the proposed statistical runtime verification method. The effects of the confidence level selection are illustrated and discussed. Higher confidence level increases the reliability and enlarges error margin for the interval. The conclusion is that the situation is inevitable and the choice of the confidence level has to be made according to the system specification. The proposed statistical runtime verification method works independently from the design flow. Thus, it can be inserted to the design flow for individual verification point or carried out outside the design flow.

We believe our attempt to statistical runtime verification to the AMS design was successful. However, the approach would be extended to online fashion without losing any accuracy and reliability in terms of confidence level and error margin. The benefits of online statistical monitoring would be: (1) interactively increase the simulation trace according to the current observed information in order to guarantee the accuracy of the results ; (2) interactively change the input in order to improve the coverage especially for the noise analysis.

References

[1] A. Aziz, K. Sanwal, V. Singhal, and R.K. Brayton. Verifying Continuous Time Markov Chains. In *Computer Aided Verification*, pages 269–276. Springer, 1996.

[2] E. Clarke, A. Donzé, and A. Legay. Statistical Model Checking of Mixed-Analog Circuits with an Application to a Third Order Δ-Σ Modulator. In *Hardware and Software: Verification and Testing*, volume 5394 of *LNCS*, pages 149–163. Springer, 2009.

[3] Collet International Research. Survey, 2002.

[4] O. Hasan. *Formal Probabilistic Analysis using Theorem Proving*. Ph.D. Thesis, Concordia University, 2008.

[5] M. Kim, S. Kannan, I. Lee, O. Sokolsky, and M. Viswanathan. Java-mac: A run-time assurance approach for java programs. *Formal Methods in System Design*, 24(2):129–155, 2004.

[6] K. Kundert. Predicting the Phase Noise and Jitter of PLL-Based Frequency Synthesizers. http://www.designers-guide.org/.

[7] P. L'Ecuyer. Uniform Random Number Generation. *Annals of Operations Research*, 53:77C120, 1994.

[8] M.P. Li. *Jitter, Noise, and Signal Integrity at High-Speed*. Prentic Hall, 2007.

[9] W. L. Martinez and A. R. Martinez. *Computational Statistics Handbook with MATLAB* . Chapman & Hall/CRC, 2001.

[10] The MathWorks. Matlab. http://www.mathworks.com/.

[11] N. Metropolis and S. Ulam. The Monte Carlo Method. *Journal of the American Statistical Association*, 39:335–341, 1949.

[12] T. Pyzdek and P. Keller. *Quality Engineering Handbook*. CRC, 2003.

[13] U. Sammapun, I. Lee, and O. Sokolsky. RT-MaC: Runtime Monitoring and Checking of Quantitative and Probabilistic Properties. In *Proc. Real-Time Computing Systems and Applications*, pages 147–153, 2005.

[14] H.L.S. Younes and R.G. Simmons. Probabilistic Verification of Discrete Event Systems Using Acceptance Sampling. In *Computer Aided Verification*, volume 2404 of *LNCS*, pages 23–39. Springer, 2002.

[15] H.L.S. Younes. Error control for probabilistic model checking. In *Verification, Model Checking, and Abstract Interpretation*, volume 3855 of *LNCS*, pages 142–156. Springer, 2006.

[16] J. Yuan, C. Pixley, and A. Aziz. *Constraint-Based Verification*. Springer, 2006.