Predictive System-Level Constraint Verification and Optimization

Andreas Krinke, Lei Lei, Jens Lienig
Dresden University of Technology, Institute of Electromechanical and Electronic Design (IFTE), Dresden, Germany
Email: andreas.krinke@tu-dresden.de

Abstract

Further automation of analog and mixed-signal integrated circuit design requires the consistent consideration of a growing number of design constraints through all design stages. However, the verification of system-level constraints is only possible towards the end of the design process when all necessary parameters are known. In this paper, we present a method for constraint state prediction in the early stages of an analog IC design project. This allows constraint consideration already during system-level design. By modeling yet unknown design parameters as random variables, the probability of a constraint to be fulfilled can be estimated. Constraint sensitivity analysis is used to identify design parameters with the most influence on a constraint's state. Finally, design parameters are optimized to maximize the probability of fulfilling all constraints.

1 Introduction

The very first step of every design process is to specify all requirements known at the time. The same applies to the design of analog and mixed-signal integrated circuits (AMS ICs). In this case, the objective is to create a circuit design that fulfills all requirements in the specification. As a first step, this set of requirements is translated into formal constraints that restrict the values of target design parameters.

Right from the start, all phases of the design process should consider these constraints to create a valid end result [1], [2]. Hence, it is necessary to calculate a constraint's state, which in turn depends on the values of all its target design parameters. However, these values are not known until later stages. As a consequence, constraint verification and consideration are not possible until these later stages are finished.

Early stages, such as system-level design, have an enormous impact on the quality of the result because corresponding design decisions influence all following design steps. In order to enable constraint consideration in these early stages, we propose a method for *constraint state prediction*. The key is to model yet unknown design parameters as random variables. Afterwards, a constraint's probability of being fulfilled can be calculated. This allows well-informed decisions in early stages despite unknown exact design parameter values.

In addition, our *constraint sensitivity analysis* ranks design parameters with regard to their impact on constraint states. This allows early identification of critical design parameters requiring particular attention later on. Finally, a novel method for *design parameter optimization* generates suggestions for design parameter values that maximize the probability of constraints being fulfilled.

After presenting related work in Section 2, we explain our constraint modeling approach in Section 3. Section 4 describes the method of predictive constraint verification, while Sections 5 and 6 give details on constraint sensitivity analysis and optimization. After presenting experimental results in Section 7, the paper ends with a summary and conclusion.

2 Related Work

Our work combines procedures from two main areas: (a) sampling of multidimensional distributions, and (b) sensitivity analysis.

The simplest method for sampling of multidimensional distributions is random sampling, where new samples are generated while ignoring all previous samples. Stratified sampling improves this approach by allowing to divide the value range in so-called stratas from which the samples are generated. This allows fine-grained control of the level of detail with which each range should be sampled. [3]

Latin Hypercube sampling is a compromise between (simple) random sampling and stratified sampling. It divides the value range automatically and creates samples for each interval. [4]

The second main area is sensitivity analysis which creates a link between changes of a model's output and changes of its inputs. On the one hand, there are methods for local sensitivity analysis, e.g., differential sensitivity analysis. Methods for global sensitivity analysis include multi-parametric sensitivity analysis (MPSA) and variance-based methods, such as Fourier amplitude sensitivity testing (FAST). [5]

3 Constraint Modeling

Constraints define requirements on design parameters. Following the approach from [6], we model a constraint c as function of design parameters that returns a Boolean value representing the state of the constraint: True if it is fulfilled, or False if it is violated.

$$c : P \rightarrow B, (p_1, p_2, \ldots, p_n) \mapsto b \qquad (1)$$

This work was supported by the German Ministry of Education and Research (BMBF) under grant 16ES03.

ISBN 978-3-8007-4444-2

Table 1 Examples of Probability Distributions

		Probability Distribution	Support supp(·)
Discrete Distributions	Finite Support	Bernoulli	$k \in \{0, 1\}$
		Binomial	$k \in \{0, \ldots, n\}$
		Degenerate	$k = k_0$
		Uniform	$k \in \{a, \ldots, b\}$
	Infinite Support	Geometric	$k \in \{1, \ldots, \infty\}$
		Logarithmic	$k \in \{1, \ldots, \infty\}$
		Negative Binomial	$k \in \{0, \ldots, \infty\}$
		Poisson	$k \in \{0, \ldots, \infty\}$
Continuous Distributions	Bounded Interval	Dirac Delta	$x = x_0$
		Kumaraswamy	$x \in [0, 1]$
		Triangular	$x \in [a, b]$
		Uniform	$x \in [a, b]$
	Semi-Infinite Interval	Chi	$x \in (0, \infty)$
		Gamma	$x \in (0, \infty)$
		Log-normal	$x \in (0, \infty)$
	Infinite Support	Laplace	$x \in (-\infty, \infty)$
		Logistic	$x \in (-\infty, \infty)$
		Normal	$x \in (-\infty, \infty)$
		Student's t	$x \in (-\infty, \infty)$

Therefore, the codomain of c is $B = \{\text{True}, \text{False}\}$. The function arguments are the constraint's target parameters $p_1, p_2, \ldots, p_n$. The values of these parameters determine the constraint's state.

Constraints can be created by logical combination of Boolean criteria. These criteria may be constructed from general, e.g. real-valued functions by introducing upper and/or lower bounds, conditions like inequality, equality, existence in a set, and so on. Equation (2) shows an example constraint on the width w and aspect ratio of an IC.

$$c(w, h) = (w < 700\,\mu\text{m}) \wedge \left(0.8 < \frac{w}{h} < 1.4\right) \qquad (2)$$

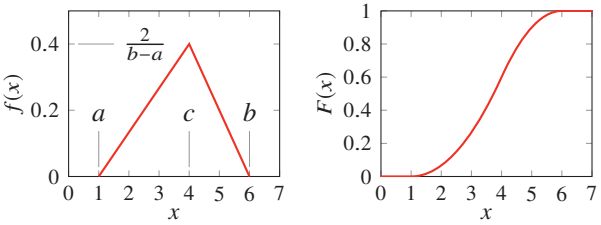
The "Global Constraint Catalog" [7] lists about 350 different constraints that may be used as criteria in a constraint function definition.

4 Constraint State Prediction

4.1 Estimation of Unknown Design Parameters

In order to determine the state of a design constraint, the values of all its target parameters have to be known. However, these values are likely to be unknown in early stages of the design process. Therefore, to predict a constraint's state early on, we model unknown design parameters as random variables to reflect this uncertainty.

Each random variable is described by some probability distribution (PD), the so-called *prior probability distribution*, or just *prior*[1]. A random variable can be (a) discrete (its PD is described by a probability mass function (PMF)), (b) continuous (its PD is described by a probability density function (PDF)), or (c) a mixture of both types. Any PD may be used to define an unknown design parameter. **Table 1** gives examples of common probability distributions and their respective supports, i.e. the set or range where the distribution is not zero-valued. Choosing a PD depends on (a) the design parameter's type: discrete or continuous, (b) the set or range where this parameter has non-zero values, and (c) the knowledge about the shape of its PMF or PDF. A simple example is a continuous design parameter for which lower bound, upper bound and mode, i.e. the value with maximum probability, can be estimated. In general, such an estimation is based on design experience. In this case, a continuous triangular distribution may be used, as shown in **Figure 1**.



(a) PDF (b) CDF

Figure 1 (a) Probability density function (PDF) and (b) cumulative distribution function (CDF) for an exemplary design parameter that was estimated using a triangular distribution with lower bound $a = 1$, upper bound $b = 6$ and mode $c = 4$. The maximum probability is $f(c) = 2 \cdot (b - a)^{-1} = 0.4$.

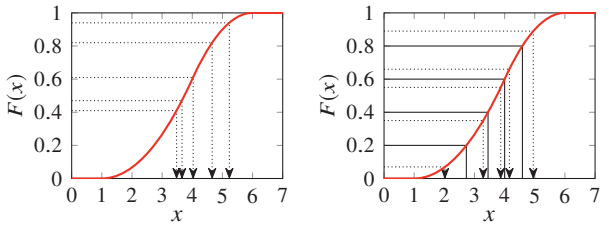
4.2 Predictive Constraint Verification

Constraint functions can be very complex. In order to predict whether or not a constraint will be satisfied later, we calculate the constraint state for large number of possible combinations of design parameter values. We use either Monte Carlo simulations (MCS) or Latin hypercube sampling (LHS) to generate a large number of random values for each design parameter's PMF or PDF.

MCS allows fast calculation of random samples for given parameter distributions. For each sample s_i, a random number r_i from a uniform distribution in the range $[0, 1]$ is sampled independently. Then, we use the inverse CDF to convert this value to a random number of the target distribution (cf. **Figure 2a**). However, depending on the number of samples, the histogram deviates from the PDF, as shown in **Figure 3a**. Ranges with low probability might be underrepresented.

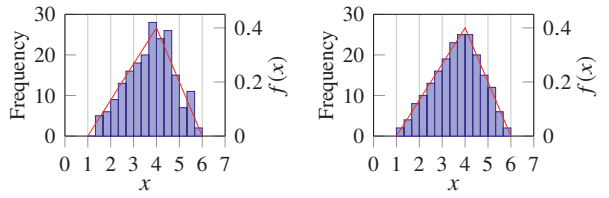
In contrast, LHS promises a more evenly sampling of the parameter distribution. The key is to divide the range of each variable in N disjoint intervals of equal probability $p = 1/N$. Using the inverse CDF, each probability interval boundary $1/N, 2/N, \ldots, (N-1)/N$ is transformed into the equivalent interval boundary within the variable range. **Figure 2b** shows these interval boundaries as solid lines. Afterwards,

[1] cf. Bayesian inference



(a) Monte Carlo sampling (b) Latin Hypercube sampling

Figure 2 Sampling of a random variable based on its cumulative distribution function (CDF, cf. **Figure 1b**) using (a) Monte Carlo sampling and (b) Latin hypercube sampling. Both methods use random numbers from a uniform distribution that are transformed using the inverse CDF. However, Latin hypercube sampling divides the variable range in intervals of equal probability at first and gets one sample from each interval.



(a) Monte Carlo sampling (b) Latin hypercube sampling

Figure 3 Comparison of the original probability density function (PDF, cf. **Figure 1a**) and the frequency distributions resulting from (a) Monte Carlo sampling and (b) Latin hypercube sampling.

one value is selected randomly from each interval exactly as in MCS. Therefore, the complete value range is covered more evenly, as shown in **Figure 3b**. [3]

In case of multiple variables, their N individual samples are combined in a random manner without replacement. For MCS and LHS, this results in N tuples, each comprising one random sample of each variable. Correlated variables are not supported in our current implementation. However, previous work describes a method for sampling random variables with a desired rank correlation matrix [3], [8]. This approach could be easily integrated.

After generating the N random parameter sets (i.e. tuples, as described above), we calculate each constraint's state (fulfilled or violated) for each sample. As a result, we can estimate the probability that a single constraint or all constraints are satisfied simultaneously.

5 Constraint Sensitivity Analysis

The goal of constraint sensitivity analysis is to calculate the sensitivity of a constraint's state to parameters. In this work we use multi-parametric sensitivity analysis (MPSA) which is a sampling-based method for *global* sensitivity analysis [9]–[11]. This method classifies the random parameter sets described in the previous section as *acceptable* or *unacceptable*. When performing MPSA for a single constraint c, a parameter set is acceptable if c is fulfilled for this set; otherwise it is unacceptable. Afterwards, we can evaluate the sensitivity to each parameter x_i by using the Kolmogorov-Smirnov two sample test [9], [12]:

$$d_{a,u}(x_i) = \sup_{x_i} |S_a(x_i) - S_u(x_i)| \tag{3}$$

with "sup" being the abbreviation for supremum, which describes the least upper bound of its argument. $S_{a,u}(x_i)$ are normalized cumulative frequency distributions:

- $S_a(x_i)$ is the distribution of the parameter samples that belong to acceptable sets, and
- $S_u(x_i)$ is the one of the samples that belong to unacceptable sets.

Therefore, $d_{a,u}(x_i)$ "can be measured directly as the greatest vertical distance between the two distribution functions plotted on the same graph" [12]. The resulting value is in the range $[0, 1]$ and represents the similarity between the two distributions. It is a measure for the sensitivity of the constraint's state to the parameter. **Figure 5** shows an example for two uniformly distributed parameters x_1 and x_2. As can be seen, x_1 has great influence, while x_2 has nearly no influence on the constraint state.

It is also possible to analyze multiple constraints by performing a logical conjunction of all constraints:

$$c_{\text{tot}} = \bigwedge_i c_i = c_1 \wedge c_2 \wedge \ldots \wedge c_i \tag{4}$$

```
1:  procedure ParameterOptimization(P, C, p_th)
2:      M ← ParameterSampling(P)          ▷ MCS or LHS
3:              ▷ calculate constraint fulfillment probability p
4:      p ← ConstraintStatePrediction(C, M)
5:      while True do
6:          v ← MPSA(P, C, M)             ▷ sensitivity analysis
7:          P' ← TranslatePDF(P, v)       ▷ move PDF l/r
8:          M ← ParameterSampling(P)      ▷ MCS or LHS
9:          p' ← ConstraintStatePrediction(C, M)
10:         if p' − p > p_th then
11:             P ← P'
12:             p ← p'
13:         else
14:             break
15:         end if
16:     end while
17:     return P
18: end procedure
```

Figure 4 Iterative algorithm for parameter optimization. The goal is to maximize the probability that all constraints are fulfilled by horizontal translation of individual parameter's probability density function (PDF) to the left or right. Arguments are the set P of all parameters, the set C of all constraints and the probability change threshold p_{th}. The set M holds tuples of parameter samples.

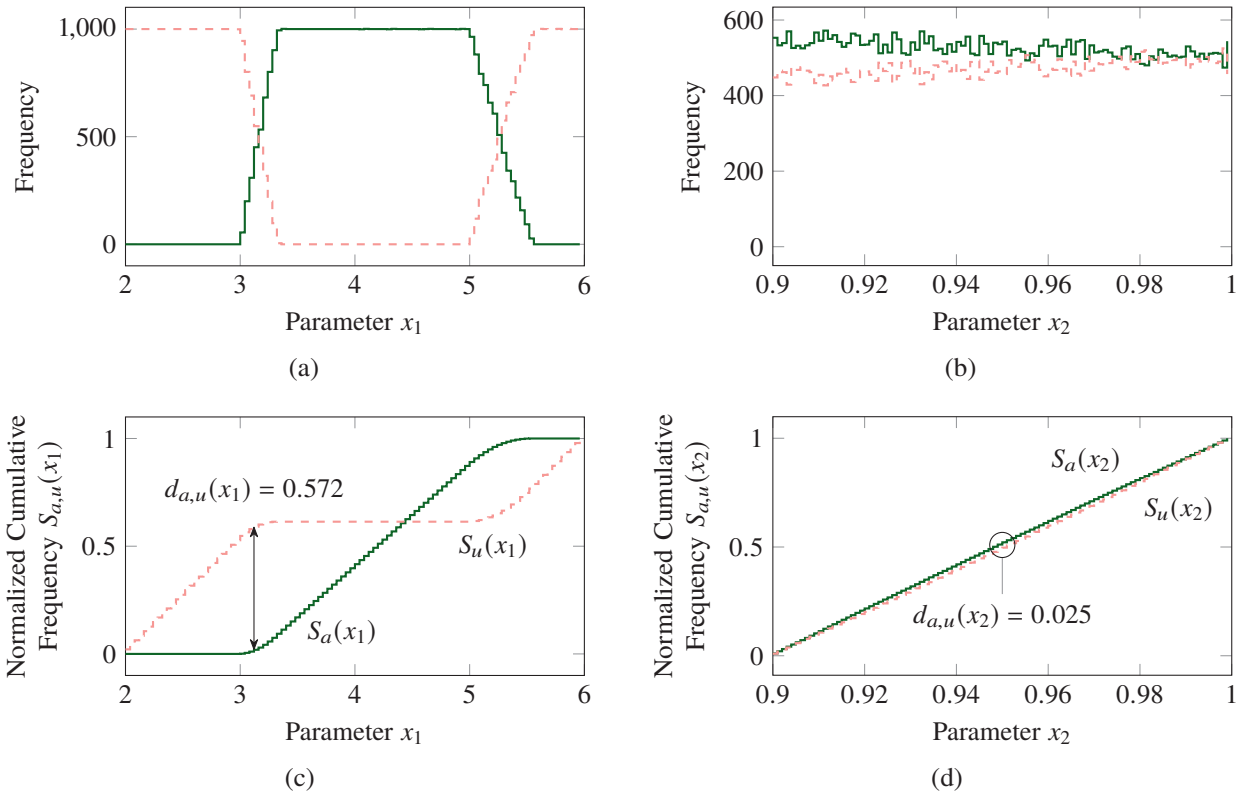


Figure 5 Frequency distributions (a, b) and normalized cumulative frequency distributions (c, d) of two parameters x_1 and x_2 (both uniformly distributed). Solid dark green lines denote frequency of (accepted) parameter samples which led to fulfilled constraints. Dashed light red lines denote frequency of (unaccepted) parameter samples which led to violated constraints. Sensitivity analysis using Kolmogorov-Smirnov two sample test calculates the maximum vertical distance $d_{a,u}(x_i)$ between normalized cumulative frequency distributions for accepted and unaccepted parameter samples $S_a(x_i)$ and $S_u(x_i)$. Due to $d_{a,u}(x_1) \gg d_{a,u}(x_2)$, the probability that all constraints are fulfilled is very sensitive to x_1 and very insensitive to x_2.

As a result, c_{tot} enforces that all other constraints c_i have to be fulfilled simultaneously. Then, sensitivity analysis can be performed as described above.

6 Design Parameter Optimization

The results of sensitivity analysis form the basis for parameter optimization. The goal is to optimize parameters in order to maximize the probability that *all* constraints are fulfilled. Our approach is to change the parameter density functions iteratively by moving them to the left or the right without changing the shape of the distribution.

Figure 4 shows the iterative algorithm for parameter optimization. We define a probability change threshold p_{th} as stop criterion for the optimization. At the beginning, we perform an initial sampling of all parameters and estimate the probability p that all constraints are fulfilled (lines 2–4). In each iteration, we use multi-parametric sensitivity analysis (MPSA) to identify the parameter v with the greatest influence on that probability p. Then, this parameter is moved to the left or the right in order to increase p. We calculate the movement direction by comparing the values of parameter v that have maximum probability of being accepted or unaccepted. In other words, we determine the position of the maxima in the frequency distributions of accepted and unaccepted values of parameter v (cf. **Figures 5a** and **5b**). If the most likely unaccepted value is greater than the most likely accepted value, we move v's distribution to the left by 10 % of that difference. Otherwise, we move the distribution to the right.

Afterwards, new samples are generated for parameter v and the probability that all constraints are fulfilled is re-estimated. In case the improvement of the constraint fulfillment probability is smaller than p_{th}, the optimization stops. The optimized parameter set P is returned.

7 Experimental Results

We implemented our algorithm using the programming language Python and the libraries NumPy, SciPy, and Numexpr [13], among others. **Figure 6** shows the main window of the graphical user interface (GUI).

The first step for the user is to enter the parameters by choosing one of the supported probability distributions and entering the corresponding values. Currently supported are: (a) normal distributions (mean μ, standard deviation σ), (b) uniform distributions (lower bound, upper bound), and (c) triangular distributions (minimum, maximum, mode).

ISBN 978-3-8007-4444-2

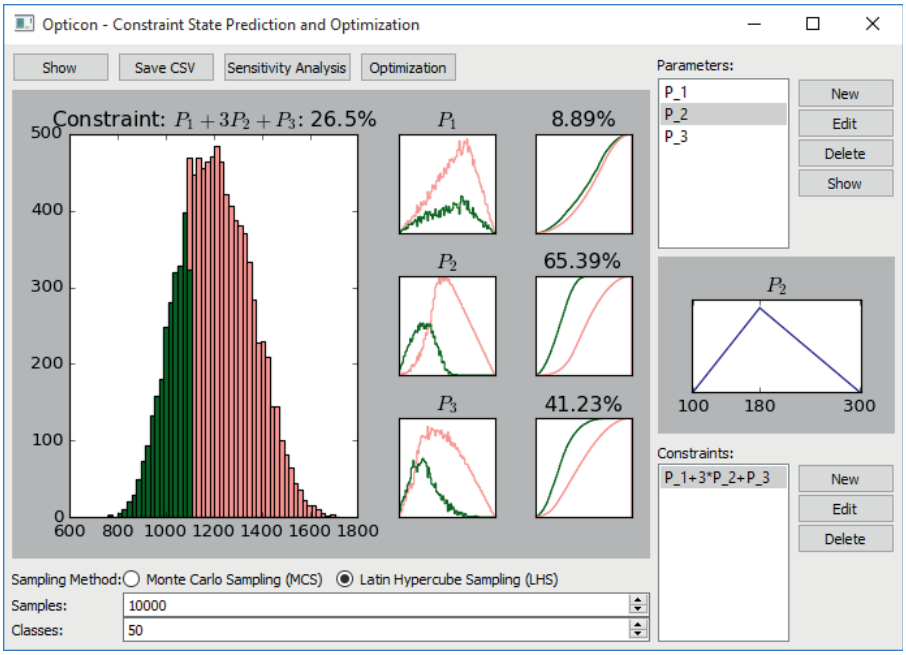


Figure 6 Screenshot of Opticon, the tool we developed for constraint state prediction and optimization. The interface for parameter and constraint definition is on the right. This example shows a constraint that limits the total power consumption of an integrated circuit containing five instances of three modules. The power consumptions P_1, P_2 and P_3 of these modules are modeled as triangular distributions. The large graph shows the distribution of the constraint function $c = P_1 + 3P_2 + P_3 < 1100$ when ignoring the limit. Samples for which c is fulfilled are highlighted in dark green. In addition, the frequency distributions and normalized cumulative frequency distributions for the accepted and unaccepted samples of P_1, P_2 and P_3 are shown (cf. **Figure 5**). The constraint's probability of fulfillment and the results of the sensitivity analysis are given as percentage above the graphs.

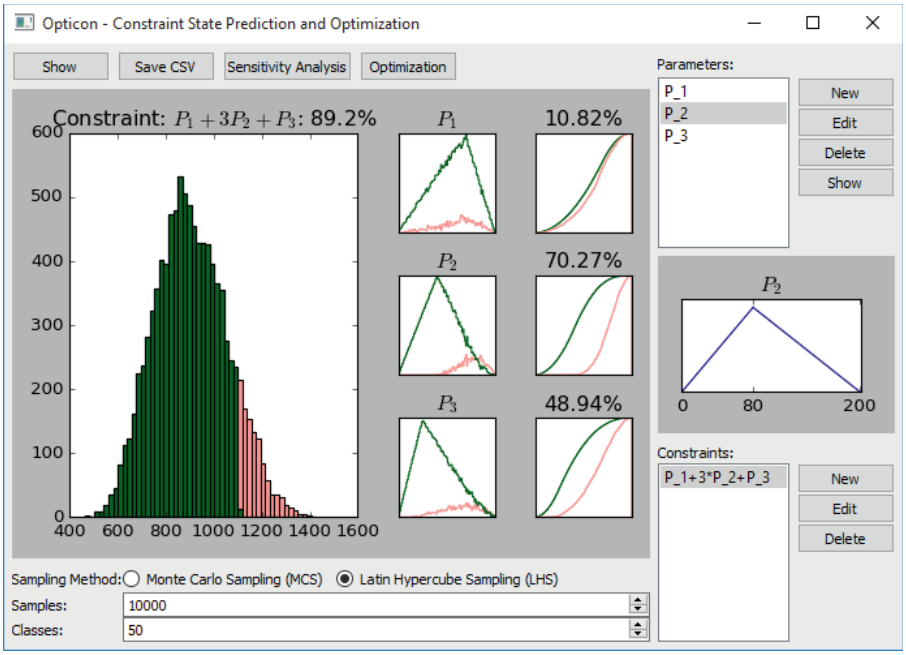


Figure 7 Screenshot of Opticon after optimization of the problem from **Figure 6**. The probability of the constraint being fulfilled increased from 26.5 % to 89.2 %. This was achieved by moving the PDF of the triangular distributed parameter P_2 to the left by $\Delta P_2 = 100$ as can be seen on the right.

ISBN 978-3-8007-4444-2

Adding new distribution types to the program is easy. In parallel, the user can decide on the sampling method and number of samples used for constraint state prediction and sensitivity analysis later on. Afterwards, constraints can be defined using these parameters.

In order to visualize the constraint state in a more meaningful way, we divide the constraint definition in two parts: (a) definition of a real-valued function, and (b) specification of a lower and/or upper bound. This second step transforms the real-valued function into a constraint that is either TRUE or FALSE—fulfilled or violated.

After parameter and constraint definition, the user is able to select one of the constraints and perform constraint state prediction. This results in a frequency distribution (histogram) of the constraint's real-valued function (on the left in **Figure 6**). The bars of the histogram are colored in dark green and light red depending on the proportion of values that fulfill or violate the constraint.

When performing sensitivity analysis, the graphs of frequency distributions and normalized cumulative frequency distributions for the accepted and unaccepted samples of all parameters are calculated and shown on the right of the constraint state prediction histogram. In addition, the sensitivity to all parameters is evaluated using the Kolmogorov-Smirnov two sample test and given as a percentage above the graphs.

Finally, parameter optimization can be performed as described in Section 6. After modification of the parameters' PDFs, all graphs are updated. **Figure 6** shows an example of a constraint that limits the total power consumption of an integrated circuit containing five instances of three modules. The parameter with the greatest influence is the power consumption P_2 of the module instanced three times. The results after optimization are shown in **Figure 7**.

8 Summary and Conclusion

In this paper, we presented a novel approach for predictive constraint verification to support well-informed decisions during system-level IC design. This addresses the problem that constraint verification is only possible after all design parameters are known. Based on a design team's experience, we model design parameters as random variables with specific probability density functions. Subsequent sampling of parameters allows us to estimate the probabilities that constraints will be fulfilled at the end of the design process.

Furthermore, constraint sensitivity analysis tells us which design parameters have the greatest influence on these probabilities. This helps to focus attention on those design aspects that are crucial for compliance with the specification.

In addition, parameter optimization gives suggestions for design parameter changes that improve probability of constraint fulfillment. Our tool Opticon gives easy access to all the aspects of this work and allows easy experimentation with different scenarios.

Future research will focus on support of correlated design parameters and investigation of variance-based sensitivity analysis methods.

References

[1] G. Jerke and J. Lienig, "Constraint-driven Design — The Next Step Towards Analog Design Automation", in *Proc. Int'l Symp. on Phys. Design*, 2009, pp. 75–82.

[2] J. Scheible and J. Lienig, "Automation of Analog IC Layout – Challenges and Solutions", in *Proc. Int'l Symp. on Phys. Design*, 2015, pp. 33–40.

[3] J. C. Helton and F. J. Davis, "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems", *Reliability Engineering & System Safety*, vol. 81, no. 1, pp. 23–69, 2003.

[4] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code", *Technometrics*, vol. 21, no. 2, pp. 239–245, May 1979.

[5] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global sensitivity analysis: The primer*. New York, NY: John Wiley & Sons, 2008, ISBN: 978-0-470-05997-5.

[6] A. Krinke, G. Jerke, and J. Lienig, "Constraint Propagation Methods for Robust IC Design", in *Proc. 8th Symp. on Reliability by Design*, ZuE, 2015, pp. 7–14.

[7] N. Beldiceanu, M. Carlsson, and J.-X. Rampon, "Global Constraint Catalog", Swedish Institute of Computer Science (SICS), Kista, Sweden, Tech. Rep. T2010:07, Nov. 2010.

[8] R. L. Iman and W. J. Conover, "A distribution-free approach to inducing rank correlation among input variables", *Communications in Statistics - Simulation and Computation*, vol. 11, no. 3, pp. 311–334, 1982.

[9] G. M. Hornberger and R. C. Spear, "An approach to the preliminary analysis of environmental systems", *Journal of Environmental Management*, vol. 12, pp. 7–18, 1981.

[10] K.-H. Cho, S.-Y. Shin, W. Kolch, and O. Wolkenhauer, "Experimental design in systems biology, based on parameter sensitivity analysis using a monte carlo method: A case study for the TNFα-mediated NF-κB signal transduction pathway", *Simulation*, vol. 79, no. 12, pp. 726–739, Dec. 2003.

[11] Z. Zi, "Sensitivity analysis approaches applied to systems biology models", *IET Syst. Biol.*, vol. 5, no. 6, pp. 336–346, Nov. 2011.

[12] D. M. Hamby, "A review of techniques for parameter sensitivity analysis of environmental models", *Environmental Monitoring and Assessment*, vol. 32, pp. 135–154, 1994.

[13] T. E. Oliphant, "Python for scientific computing", *Computing in Science Engineering*, vol. 9, no. 3, pp. 10–20, May 2007.