Monte Carlo Simulation with Modelica

Joachim Haase Susann Wolf Christoph Clauß
Fraunhofer-Institute for Integrated Circuits, Design Automation Division
Zeunerstraße 38, 01069 Dresden, Germany
{Joachim.Haase, Susann.Wolf, Christoph.Clauss}@eas.iis.fraunhofer.de

Abstract

Monte Carlo simulation allows to obtain statistical information derived from estimates of the random variability of component parameters. The paper demonstrates how to describe the random characteristic of parameters in a tool-independent manner in Modelica. Using the multi-run facilities of a simulation engine statistical analysis can be carried out without any code intervention concerning the tool. The approach is based on the SAE 2748 standard. Solutions of implementation problems with respect to Modelica are discussed. This paper is based on results, which were developed in the Fraunhofer collaborative project "Computer Aided Robust Design (CAROD)".

Keywords: Statistical analysis, SAE 2748, Monte Carlo simulation

1 Introduction

It is more and more required within industrial applications to consider the influence of the variability of design parameters on the behaviour of systems. For instance yield and reliability often depend on the statistical characteristics of such parameters [1].

Monte Carlo methods are widely used to analyze the effects of parameter tolerances. In a Monte Carlo simulation, a mathematical model of a system is repeatedly evaluated. Each run uses different values of design parameters. The selection of the parameter values is made randomly with respect to given distribution functions. Monte Carlo simulation is very time consuming. A lot of simulation runs are required to investigate the behavior of a system subject to the statistical distribution of parameters. Nevertheless, Monte Carlo simulation is very favored in various application areas where an analytical relation between design and system parameters is difficult to find. For example mixed-signal electrical systems consisting of analog and digital components often belong to this class of systems.

The objective of this paper is to make a proposal how to handle the description of random parameters in Modelica in a tool-independent way. Furthermore a way is presented how to carry out a Monte Carlo simulation within an existing simulation engine. It is only required that the simulator supports multiple runs of a simulation task.

The approach is close to the standard J 2748 prepared by the Electronic Design Automation Standards Committee of the Society of Automotive Engineers (SAE) that describes random parameter handling in a VHDL-AMS simulation problem [2, 3]. Describing parameter variations in nearly the same way in VHDL-AMS and Modelica offers the opportunity to reduce the effort to provide random parameter data in the design process and to avoid misunderstandings.

2 SAE-Standard J 2748

Some basic requirements that are supported by the SAE J 2748 standard are summarized in the following. The basic idea is to add information to characterize the parameters. Thus, it should be possible to use existing models also for statistical analysis. In detail it is required

- Usage of the same model for nominal and Monte Carlo analysis
- Possibility to assign different statistical distributions to each constant or parameter
- Support of continuous and discrete distributions
- Permission of user-defined distributions
- Possibility to specify correlation between constants

From a practical point of view the following points should also be mentioned

- Independent random number generation for any constant

- Reproducibility of Monte Carlo simulation within the same simulation tool

Statistical distributions are characterized from an engineering point of view. That means the mathematical parameters as for instance the moments are derived from engineering parameters as nominal value, tolerances, minimum and maximum values. The standard provides implementations of basic regular distribution functions. Futhermore, standard functions are provided that allow to declare user-defined distributions. Also truncated distributions are supported that limit the random numbers to a given interval.

Table 1: Regular distribution functions [3]

UNIFORM	Uniform distributed values
NORMAL	Gaussian distributed values
PWL_CDF	Piecewise-linear description of a cumulative distribution function
PWL_PDF	Piecewise-linear description of a probability density function
BERNOULLI	Bernoulli distribution
DISCRETE_CDF DISCRETE_PDF	Tabular description of the probability of discrete values

The VHDL-AMS implementation details are online available [4].

3 Method

Methods to create random numbers are in general based on a (0,1) uniform distributed values.

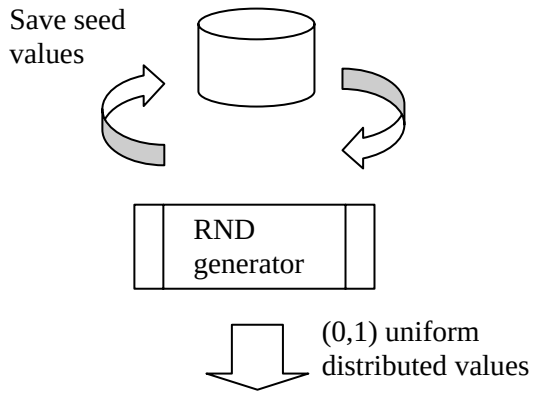

Fig. 1. (0,1) uniform random number generator

Widely used methods to generate random number with a given distribution are the inverse transformation approach based on the cumulative distribution function and its modifications for truncated distributions. The Box-Muller algorithm can be applied for normal distributed numbers [5].Thus, the main problem during parameter initialization for Monte Carlo Simulation is to generate independent (0,1) distributed values. [2] describes the requirements to a built-in random number generator provided by a tool.

The basic idea of a tool-independent random number generator is shown in Fig. 1. The seed values that are needed to generate a sequence of random numbers are immediately saved in a file.

With the help of global parameters it is possible to switch between nominal and statistical analysis either w.r.t. parts of a description or the entire simulation task.

4 Realization with Modelica

Using Modelica the idea of a tool independent random number generation is realized in the following way. As an example the uniform distribution is used which produces uniformly distributed values within the interval (nominal – tolerance*nominal, nominal + tolerance* nominal). For better reading some details compared to the final solution are simplified.

4.1 Randomly changed parameters

To supply a parameter (or a constant) with randomly generated values it is necessary to specify random distribution in the Modelica source code. Instead of

```
parameter Real p = nominal;
```

which specifies a fixed parameter, the specification of the uniform distribution function call is:

```
parameter Real p = uniform(nominal,
                           tolerance);
```

4.2 Random number generation

The Modelica function uniform is an interface to a C function. It is defined like this:

```
function uniform
  input Real Mean;
  input Real Tol;
  output Real random_value;
external "C" uniform(Mean, Tol,
                     random_value);
end uniform;
```

Within the C function the randomly distributed values have to be calculated. An example is the following function:

```
void uniform (double M, double Tol,
              double *aus)
{ double xMin = M * (1.0 - Tol);
  double xMax = M * (1.0 + Tol);
  if (xMin > xMax)
  { xMax = xMin;
    xMin = M * (1.0 + Tol);
  }
  *aus = xMin + (xMax - xMin)*RND();
}
```

The random function is a (0,1) uniformly distributed random value generator for instance according to Schrages method [8]:

```
double RND()
{ FILE   *read_fp, *write_fp;
  long   seed = 2,  M = 2147483647;
  long   A = 16807, Q = 127773;
  long   R = 2836,  k;
  double F = 1.0/ M;

  read_fp = fopen ("seed.dat","r");
  fscanf (read_fp, "%ld",&seed);
  fclose(read_fp);

  assert( seed != 0 );
  k = seed / Q;
  seed = (seed - k * Q) * A - k * R;
  if ( seed < 0 ) seed += M;

  write_fp = fopen ("seed.dat","w");
  fprintf (write_fp, "%ld", seed);
  fclose(write_fp);

  return seed * F;
}
```

By access to the file “seed.dat” , which has a fixed name, the seed value is saved between two calls of the random function.

In the final solution a global change of the seed file name is possible. In case of a nominal analysis the final function uniform would deliver the Mean value. A more convenient way would be to provide the random number generator RND by a Modelica function. This would allow to formulate the random distribution functions using Modelica language constructs only. This approach could not be realized in the used tool environment. From the language point of view it must be possible that a Modelica function called with the same arguments may deliver different results. For this reason, for instance VHDL(-AMS) distinguishes between pure and impure functions.

Furthermore, the RND function above could be replaced by the random number generator incorporated in a Modelica simulator by a tool provider. In this way the file access to seed.dat can be avoided.

4.3 Application

After having specified the parameter to be changed in the Modelica source code, the Modelica function with the foreign function interface to the C domain, and the C function “random”, the following steps are necessary:

A file “seed.dat” has to be generated, which contains an integer starting number for the sequence of random values. If a sequence shall be repeated, the same seed number must be chosen.

Then the model under investigation (which contains the parameter specification mentioned above) has to be simulated by a Modelica simulator repeatedly. The number of repetitions depends on the wanted number of trials for the Monte Carlo simulation. After each single simulation the interesting results must be saved. The results can be visualized or used in posteriori calculations.

4.4 Remarks

The method allows easily to define both correlated and dependent random values of parameters. A simple example might explain the procedure:

```
parameter Real p1 = uniform(1, 0.1);
parameter Real p2 = uniform(p1, 0.01);
```

If the same sequence of randomly generated values is desired (e.g. to investigate a special effect) the same seed number and the same seed file name have to be used at the beginning.

5 Example

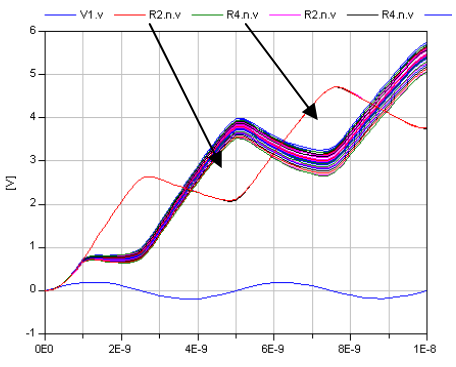

Fig. 2. Monte-Carlo-Plot for variables

In the DifferenceAmplifier of Modelica.Electrical. Analog.Examples [9]. the resistance R of the resistor R2 is randomly generated by the following formulation:

```
...
Basic.Resistor R1(R=0.0001);
Basic.Resistor R2(R=uniform(100,0.05));
Basic.Resistor R3(R=0.0001;
...
```

Repeated simulations using Dymola show that R2.n.v (the thick line pencil) is sensitive with respect to R2.R. The voltage R4.n.v (thin line) is not sensitive to that parameter. Basing on the Monte Carlo results further calculations (density distribution …) are possible.

Furthermore, the randomly chosen parameter values can also be visualized or used for further calculations. The following figure shows the above specified parameter R2.R which is uniformly distributed in the interval (95, 105) (=100 – 100 * 5%, 100 + 100 * 5%).

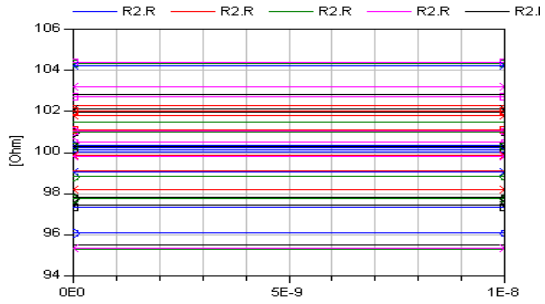

Fig. 3. Randomly chosen parameter R2.R

6 Discussion

The proposed approach realizes a simple Monte Carlo simulation based on behavioral descriptions in Modelica. Beyond the focus of this paper is the usage of the results of the Monte Carlo simulation for other purposes. For example the data could be used to create Response Surface Models. This would require to save the randomly generated parameters of any simulation run. Also improved techniques to create the random numbers and reduce the simulation effort could be applied. For instance possibilities of so-called importance sampling [6] could be applied using user defined functions.

The Monte-Carlo-Simulation is also possible using the Dymola Monte-Carlo feature. The advantage of the suggested way is:

- It is a more general, tool independent approach.
- The user is free to define its own distribution based on the RND function.
- Correlations can be defined easily.
- For documentation purposes the distribution specification is part of the model files.

The approach in [7] is also simulator independent, but is uses a (firm-)specific nested toolkit. Our way is defined only using the Modelica language. Whether a language construct like ours is used in [7] is not documented.

7 Conclusions

An approach to handle statistical analysis problems within Modelica is presented. It is based on the SAE J 2748 standard. The current version allows Monte Carol simulations if the used simulation engine supports multiple runs in a simple way. If the approach is accepted it could also be the basis of efficient implementation in Modelica simulators. In this case the generation of the sequences of (0,1) distributed uniform random numbers must be supported without file access.

The applicability of the approach is demonstrated with the help of a simple example from the existing Modelica standard library. Only existing tool and language features are used. This and the orientation to the SAE standard are the main advantages of the approach compared to [7].

References

[1] O'Connor, P.D.: *Practical Reliability Engineering.* John Wiley & Sons, 2003 (5th ed.)

[2] J2748, *VHDL-AMS Statistical Analysis Packages*, The SAE Electronic Design Automation Standards Committee, Troy, MI, October 2006

[3] Christen, E.; Bedrosian, D.; Haase, J.: *Statistical Modeling with VHDL-AMS*. Proc. Forum on Specification and Design Laguages FDL '07, Barcelona, September 18-20, 2007.

[4] http://links.sae.org/j2748
http://fat-ak30.eas.iis.fraunhofer.de

[5] Saucier, R. *Computer Generation of Statistical Distributions*. US Army Research Lab ARL-TR-2168, available at http://ftp.arl.mil/random/random.pdf

[6] Robert, C. P.; Casella, G.: Monte Carlo Statistical Methods. Springer, 2004 (2nd ed.)

[7] Batteh, J.; Tiller, M.; Goodman, A.: Monte Carlo Simulations for Evaluating Engine NVH Robustness. 4th International Modelica Conference, Hamburg, March 7-8, 2005, 385-392

[8] www.physics.rutgers.edu/grad/509/random.pdf

[9] www.modelica.org/libraries/Modelica