VHDL-AMS Statistical Analysis for Marginal Probabilities

Usage of SAE J 2748 Statistical Analysis Package for Importance Sampling

Joachim Haase
Branch Lab Design Automation (EAS)
Fraunhofer-Institut Integrierte Schaltungen
Dresden, Germany
joachim.haase@eas.iis.fraunhofer.de

Christoph Sohrmann
Branch Lab Design Automation (EAS)
Fraunhofer-Institut Integrierte Schaltungen
Dresden, Germany
christoph.sohrmann@eas.iis.fraunhofer.de

Abstract— **The impact of parameter variations on components' and systems' characteristics, especially in the area of IC design, has been discussed for several years. To investigate the influence of parameter variations on system characteristics, standard Monte Carlo simulation is often used when exact results cannot be obtained using a deterministic algorithm. However, this procedure may require a huge number of simulation runs if marginal probabilities are estimated. This paper shows how importance sampling as a variance reduction technique can be used for estimating small probabilities in simulation experiments based on the SAE J 2748 VHDL-AMS Statistical Analysis Package. Furthermore, application examples are presented to show how the use of parameter sensitivities can help creating random variable distributions for importance sampling.**

Keywords: Monte Carlo simulation, importance sampling, VHDL, VHDL-AMS, yield analysis

I. INTRODUDUCTION

The impact of parameter variations on components' and systems' characteristics, especially in the area of IC design, has been discussed for several years [1]. In this paper, we focus on one approach to handle variations that lead to marginal violation of design objectives. That is, we are interested in cases where the probability that such a violation occurs is small.

In order to investigate the influence of parameter variations on system characteristics, Monte Carlo simulation is often used when exact results cannot be obtained using a deterministic algorithm. This procedure allows, for instance, for estimating probability densities of targets using repeated computations with random parameters. However, the number of required simulation runs can increase dramatically if small probabilities have to be estimated that describe the violation of design objectives.

The standard Monte Carlo procedure is well known. Repeated simulation runs are carried out using different random parameters for each run. The number of successive runs divided by the total number N of simulation runs estimates the probability P of the occurrence of the event under investigation. The successively estimated values form a Gaussian-distributed random variable with the following standard deviation

$$o^2 = \frac{1}{N} \cdot \left(P - P^2 \right) \qquad (1)$$

Thus, with a probability of 95 %, the final value that has to be estimated lies in the interval $\left[P - 2\sigma, P + 2\sigma \right]$. An equivalent characteristic can easily be derived. The number of standard Monte Carlo simulation runs to assure that the value P belongs to the 95 % confidence interval $\left[P - \alpha P, P + \alpha P \right]$ must meet the inequality [2]

$$N \geq \frac{4}{\alpha^2} \cdot \frac{1 - P}{P} \qquad (2)$$

Equation (2) demonstrates the problem if P is small. If we accept a value $\alpha = 0.1 = 10\%$ and expect, for instance, a marginal probability P of 10^{-4}=0.1 ‰ it follows $N > 4 \cdot 10^6$. Thus, nearly half a million of simulation runs are required, which, in most cases, is unacceptably high. On the other hand, equation (2) delivers for a fixed number N the limits of the confidence interval.

One method to overcome this problem is importance sampling. The samples are generated in a special way that reduces the variance of the estimator of P compared to standard Monte Carlo simulation for the same number N. Importance sampling has been known for a long time [3, 4, 5]. The main question is how to choose an appropriate sampling distribution for a given problem. The other question is how to apply the method using existing simulation tools.

Describing the statistical behavior of the random variables and their correlations and questions regarding the accuracy of the models under investigation are beyond the focus of this paper. We concentrate on importance sampling as a variance reduction technique to generate simulation experiments, which reduces simulation efforts.

In the following, we start with some basics of importance sampling and introduce a procedure to implement the method within VHDL/VHDL-AMS simulation tools based on the SAE J 2748 VHDL-AMS Statistical analysis package. Afterwards, we present procedure that determines the probability distributions for importance sampling in the case of Gaussian-distributed parameters. We exploit parameter sensitivities in order to derive these functions. Finally, we demonstrate the use of this method for delay and power analysis of two simple circuits.

II. IMPORTANCE SAMPLING BASICS

A. General Approach

Importance sampling technique is widely discussed in the mathematical literature [3, 4, 5] and some technical papers as well [2, 6, 7]. We summarize the basic results we will need subsequently. Let p be a scalar probability density distribution. Then, the probability that the associated random variable is greater than the value T is given by

$$I = \int_T^\infty p(x)dx \tag{3}$$

The idea is now to evaluate instead of (3) the equation [3]

$$I = \int_T^\infty \frac{p(x)}{g(x)} \cdot g(x)dx \tag{4}$$

if g is used instead of p as trial distribution. To estimate I, random values distributed with the probability density function g are generated. Thus, I can be estimated by

$$\tilde{I} = \frac{1}{N} \cdot \sum_{k=1}^{n} \delta(x_k) \cdot \frac{p(x_k)}{g(x_k)} = \frac{1}{N} \cdot \sum_{k=1}^{N} \delta(x_k) \cdot w(x_k) \tag{5.1}$$

$$\text{with} \quad \delta(x_k) = \begin{cases} 0 & for \quad x_k \le T \\ 1 & for \quad x_k > T \end{cases} \tag{5.2}$$

The w values are called weights. Instead of the unbiased estimator (5), especially for smaller N, the more accurate estimator should be applied [3]

$$\hat{I} = \frac{\sum_{k=1}^{n} \delta(x_k) \cdot \frac{p(x_k)}{g(x_k)}}{\sum_{k=1}^{N} w(x_k)} = \frac{\sum_{k=1}^{N} \delta(x_k) \cdot w(x_k)}{\sum_{k=1}^{N} w(x_k)} \tag{6}$$

The standard deviation σ_I of the estimator can be determined by [6]

$$\frac{\sigma_I}{I} = \frac{1}{\sqrt{N}} \cdot \left(\frac{I_2}{I^2} - 1 \right)^{\frac{1}{2}} \tag{7}$$

$$\text{with} \quad I_2 = \int_T^\infty \frac{p^2(x)}{g(x)} dx \quad \text{and} \quad I = \int_T^\infty p(x)dx$$

For a great number N, we estimate the associated variance based on the simulation results using g

$$\sigma_I{}^2 \approx \frac{1}{N} \cdot \left(\frac{1}{N} \cdot \left(\sum_{k=1}^{N} \delta(x_k) \cdot w(x_k)^2 \right) - \hat{I}^2 \right) \tag{8}$$

It is evident that if the distribution function g equals p, the formulas (4) to (8) are equivalent to standard Monte Carlo simulation results.

The main and difficult task is to find a good distribution g that can be applied in importance sampling. In theory, the best g is given by [3, 4]

$$g^*(x) = \frac{\delta(x) \cdot p(x)}{\int_T^\infty p(x)dx} \tag{9}$$

This is of little practical interest because the value I we want to estimate is needed as denominator in (9). But what we see is that the shape of g should be near $\delta(x) \cdot p(x)$ [4].

Importance sampling by scaling and translation are widely used with the density functions

$$g(x) = \frac{1}{a} p\left(\frac{x}{a} \right) \tag{10}$$

and

$$g(x) = p(x - c) \tag{11}$$

respectively. Of practical importance is the usage of a mixed density function using the original distribution p and r (at least one) other distributions $h_i(x)$ [5]

$$g(x) = \left(1 - \sum_{i=1}^{r} \lambda_i \right) \cdot p(x) + \sum_{i=1}^{r} \lambda_i \cdot h_i(x) \tag{12}$$

$$\text{with} \sum_{i=1}^{r} \lambda_i < 1 \quad \text{and} \quad \forall_{1 \le i \le r} \quad \lambda_i \ge 0$$

In the following, we will use a special case of (12). Let p be the density of a $N(m, \sigma)$ Gaussian distributed parameter and g given by a combination of p and a translated distribution p. Thus, a special case of (12) is given by

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} \left((1 - \lambda) \cdot e^{-\frac{(x-m)^2}{2\sigma^2}} + \lambda \cdot e^{-\frac{(x-m_{is})^2}{2\sigma^2}} \right) \tag{13}$$

$$\text{with} \quad m_{is} = m + c$$

B. Application scenario

We assume that the q parameters of interest, that may vary and influence the system, are independent and Gaussian distributed. Otherwise, we have to start with a PCA in order to transfer the original problem into a problem with independent alternative parameters.

At the beginning of each simulation run, the parameters are chosen with respect to (13); that is, with distributions p_j and g_j with associated m, σ, m_{is}, and λ. The weight of a simulation run is given by

$$w(x) = \frac{p(x)}{g(x)} = \prod_{j=1}^{q} \frac{p_j(x_j)}{g_j(x_j)} \tag{14}$$

δ indicates whether the value of the objective is greater T or not. Using (6) and (8) we can analyze the simulation results.

III. IMPLEMENTATION IN VHDL-AMS/VHDL

A. SAE J 2748 Statistical Analysis Package

The Statistical Analysis Package provides VHDL-AMS functions that can be used for describing the random behavior of parameters in a VHDL/VHDL-AMS description [8, 9]. At the beginning of each simulation run the parameters are initialized using random values distributed in accordance with the associated probability density or cumulative density function (pdf or cdf respectively).

The fundamental function described by the SAE J 2748 standard is a random number generator STD_UNIFORM that delivers (0, 1) distributed uniform numbers. The uniform random numbers can be transformed with respect to the required distribution function. The idea behind STD_UNIFORM is to access a random number generator for uniform values as it is given in the MATH_REAL package of the IEEE library by the UNIFORM procedure. The SEED values of this procedure UNIFORM must be handled using a global storage place. This place can be without further requirements to the simulator a file. The function STD_UNIFORM reads the SEED values from this file and determines the next (0, 1) distributed value calling the UNIFORM procedure that also delivers new SEED values. These new SEED values are written to the file and are used during the next activation of STD_UNIFORM.

Thus, every VHDL-AMS/VHDL simulator that allows for multiple runs can be used for setting up standard Monte Carlo experiments. It is also possible to replace the file by handling the administration of global SEED places inside a simulator.

TABLE I. SELECTED FUNCTIONS PROVIDED BY SAE J2748

Function name	*Comment*
STD_UNIFORM	Delivers a random value between 0 and 1
STD_NORMAL	Delivers a N(0,1) distributed random value
NORMAL	Delivers a Gaussian distributed random value with given mean value and standard deviation described by tolerances of min/max limits.
BERNOULLI	Delivers Bernoulli distributed random numbers.
PDF	Delivers a random value described by a piecewise linear description of a pdf
CDF	Delivers a random value described by a piecewise linear description of a cdf

Some of the available functions are shown in Table 1. PDF and CDF are also provided for discrete random numbers. These functions can be used for initializing constants in VHDL/VHDL-AMS descriptions or they can be used for assigning random values to existing models during the instantiation.

The functions are provided in a package STATISTICS of a library named VHDL_UTILITY. The functions of this package can also be used for creating user-defined random number generators. This way, specific functions that can be used for importance sampling can be declared.

B. Functions for importance sampling

The STATISTICS package can be used for declaring a function IS_NORMAL in accordance with (13) with the parameters MEAN=m, SIGMA=σ, LAMBDA=λ, and MEAN_IS=m_{is}. The function header of the associated VHDL function is shown in Figure 1.

IS_NORMAL

```
impure function IS_NORMAL (
      MEAN: REAL := 0.0,
      SIGMA: REAL := 1.0,
      LAMBDA: REAL := 0.5,
      MEAN_IS: REAL := 0.0,
      MODE: STAT_MODE_TYPE
      ) return REAL;
```

Argument	Type	Default	Description
MEAN	REAL	0.0	mean value of original distribution
SIGMA	REAL	1.0	standard deviation of original distribution
LAMBDA	REAL	0.5	contribution of original pdf after translation
MEAN_IS	REAL	0.0	mean valued of original pdf after translation
MODE	STAT_MODE_TYPE	STAT_MODE	activates nominal or statistical evaluation
RETURN	REAL		random number for importance sampling for MODE /= STAT_NOMINAL

Dependencies

- VHDL_UTILITY.STATISTICS.ALL
- VHDL_UTILITY.STATISTICS_CONTROL.ALL

Figure 1. Description of function IS_NORMAL

Let x_j be the random value that is determined based on (13), then the associated weight $\frac{p_j(x_j)}{g_j(x_j)}$ used in (14) can also be determined within IS_NORMAL. The product (14) can be updated in an analog way as described in the previous subsection for the function STD_UNIFORM. Before the start of a simulation run, the product of weights must be initialized with 1. This value is saved in a file for instance. The product is updated at the end of each call of the function IS_NORMAL. The function is declared as shown in the following:

```
library VHDL_UTILITY, STD;
use VHDL_UTILITY.STATISTICS.all;
use VHDL_UTILITY.STATISTICS_CONTROL.all;

package body IS_STATISTICS is
...

impure function IS_NORMAL
    ( MEAN        : REAL := 0.0;
      SIGMA       : REAL := 1.0;
      LAMBDA      : REAL := 0.5;
      MEAN_IS     : REAL := 0.0;
      MODE        : STAT_MODE_TYPE := STAT_MODE) return REAL
is
     variable VALUE        : REAL;
     variable RATIO        : REAL;
     variable RND          : REAL;
begin
 assert SIGMA > 0.0
 report "SIGMA must be > 0.0.";

 assert LAMBDA >= 0.0 and LAMBDA <= 1.0
 report "LAMBDA between 0.0 and 1.0 required.";
```

```
if MODE = STAT_NOMINAL then
 VALUE := MEAN;
 RATIO := 1.0;
else
 if LAMBDA = 0.0 then
  VALUE := VHDL_UTILITY.STATISTICS.NORMAL
           (MEAN,MEAN-SIGMA,MEAN+SIGMA,FALSE,1.0);
  RATIO := 1.0;
 elsif LAMBDA = 1.0 then
  VALUE := VHDL_UTILITY.STATISTICS.NORMAL
           (MEAN_IS,MEAN_IS-SIGMA,MEAN_IS+SIGMA,FALSE,1.0);
  RATIO := EXP(-((VALUE-MEAN)/SIGMA)**2/2.0)/
           EXP(-((VALUE-MEAN_IS)/SIGMA)**2/2.0);
 else
  RND := VHDL_UTILITY.STATISTICS.STD_UNIFORM;

  if RND < LAMBDA then
   VALUE := VHDL_UTILITY.STATISTICS.NORMAL
            (MEAN,MEAN-SIGMA,MEAN+SIGMA,FALSE,1.0);
  else
   VALUE := VHDL_UTILITY.STATISTICS.NORMAL
           (MEAN_IS, MEAN_IS-SIGMA,MEAN_IS+SIGMA,FALSE,1.0);
  end if;
  RATIO := EXP(-((VALUE-MEAN)/SIGMA)**2/2.0)/
           ( (1.0-LAMBDA)*EXP(-((VALUE-MEAN)/SIGMA)**2/2.0)
            +LAMBDA*EXP(-((VALUE-MEAN_IS)/SIGMA)**2/2.0) );
  end if;
 end if;
 UPDATE_RATIO(RATIO);  -- update of weight product
return VALUE;
end function IS_NORMAL;

end package IS_STATISTICS;
```

The update of the weights as described by equation (14) is carried out using the function UPDATE_RATIO. The function reads the old weight product from a file, multiplies it with the RATIO value, and writes it back to a file. Using the same file that carries the current weight product, IS_NORMAL can also be used together with other functions. Thus, at the end of the initialization phase, the associated file saves the product. At the end of each simulation run this product and the value of the objective function have to be saved and can later be used to analyze the results using equations (6) and (8).

Example: A constant PARAMETER with mean value 2.0, standard deviation 0.4, mean value 3.2 after translation, and $\lambda = 0.5$ can be initialized in the following way:

```
constant PARMETER : REAL
:=IS_NORMAL(MEAN=>2.0, SIGMA=>0.4, MEAN_IS=> 3.2);
```

At the end of each simulation run the value of the objective function and the weight that was determined in the parameter initialization phase must be written to a file. Afterwards, the file with the product of weights must be reinitialized with 1 for the next simulation run. The functions WRITE_IS_RESULTS, READ_WEIGHTS, and RESET_WEIGHT are declared to carry out these actions.

Example: After determining the objective value VALUE, the following statements should be carried out

```
WEIGHT := READ_WEIGHT;
WRITE_RESULTS (WEIGHT, VALUE, "results.dat");
RESET_WEIGHT;
```

IV. EXAMPLES

A. Parametrization of functions for importance sampling

The crucial task in importance sampling is to choose an appropriate function g (see section II.). In practical applications, the best g cannot in general be determined analytically. Hence, we try to determine the parameters of a predefined function by minimizing the variance of the estimated values using the equation (7). We assume Gaussian distributed parameters in the following.

If p describes a N(0,1) distributed Gaussian variable and g is given by $g(x) = p(x-c)$ we have to minimize (based on (7))

$$H(c,T) = \frac{\frac{1}{\sqrt{2\pi}} \cdot \int_T^\infty e^{c^2} \cdot e^{-\frac{(x+c)^2}{2}} dx}{\left(\frac{1}{\sqrt{2\pi}} \cdot \int_T^\infty e^{-\frac{x^2}{2}} dx \right)^2} - 1 \rightarrow \min \qquad (15)$$

c has to be determined depending on the limit T used in (3). Figure 2 depicts H. We obtain a minimum for c = 1.3 T… 1 T depending on T and should mention that this result can be transferred to N(0, σ) distributed variables.

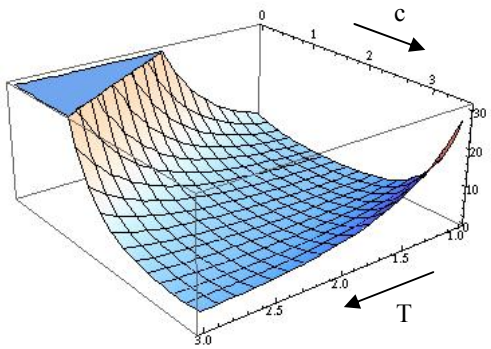

Figure 2. H(c, T) for c = 0 …3 and T = 1 … 3

Based on this experience, we try to motivate a heuristic approach to determine the delay c used in (13). Let *Y(P)* be the objective value we are interested in depending on the parameter *P*. The nominal parameter value shall be P_0. We are interested in the probability that Y is greater than T (where $Y(P_0) = Y_{no\min al} < T$). Using first-order parameter sensitivities we approximate

$$Y(P) \approx Y(P_0) + \frac{\partial Y}{\partial P} \cdot \Delta P$$

and require

$$Y_{no\min al} + \frac{\partial Y}{\partial P} \cdot c = T \qquad (16)$$

Equation (16) allows now to determine c. In the multivariate case we get for c_j (used instead of c for the description of parameter P_j)

$$Y_{nominal} + \sum_{j=1}^{r} \frac{\partial Y}{\partial P_j} \cdot c_j = T \qquad (17)$$

Among the potential solutions we choose the one that delivers the greatest probability for $x_j = c_j$ in order to be "near" the shape of the "best" function g^* (see section II. subsection A.)

$$\prod_{j=1}^{q} p(c_j) \rightarrow \max \qquad (18)$$

For Gaussian distributed values we can try to evaluate

$$\sum_{j=1}^{r} \frac{{c_j}^2}{{\sigma_j}^2} \rightarrow \min \qquad (19)$$

Thus, (17) and (19) establish requirements for the determination of c_j. We used this approach in the subsequent examples. Another possibility based on the so-called center of gravity (C.O.G.) is discussed in [2].

B. *Gaussian distribution*

For test purposes we start with a N(m, σ) distribution with *m=2* and $\sigma = 0.4$. We are interested in the probability that the random variable *X* is greater than $T = 3.2 = m + 3\sigma$. The expected probability is $\frac{1}{2} \cdot \left(1 - erf\left(\frac{3}{\sqrt{2}}\right)\right) \approx 1.3499 \cdot 10^{-3}$.

We use the VHDL function call

```
IS_NORMAL(MEAN=>2.0, SIGMA=>0.4, MEAN_IS=> 3.2)
```

to estimate the probability.

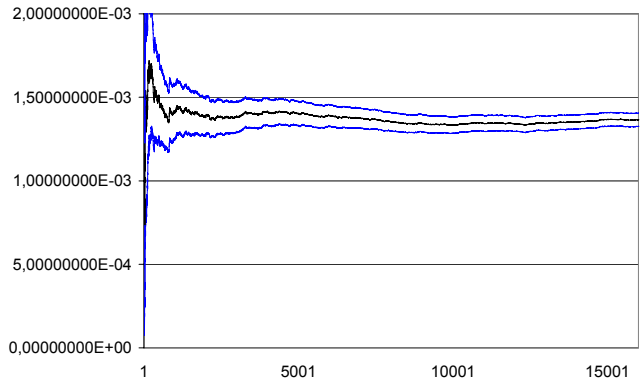

Figure 3. Estimated values for Prob(X>3σ) depending on the nmber of simulation runs

Figure 3 shows the progress of the estimated values depending on the number of simulation runs. Based on the variance estimated using (8), the limits of the 95 % confidence interval are added to the diagram.

C. *Inverter*

We determine the input/output delay of a simple inverter. Parameters and description are based on a 90nm technology (VDD=1.22V) [10]. The output is connected by a capacitance of 0.03 pF to the reference. We assume Gaussian distributed parameters with a standard deviation of 10 % of the nominal values (based on [11], Table 6.1).

The nominal values of the considered transistor parameters are PCHWIDTH=3.8U, NCHWIDTH=2.6U, PCHL=0.28U, NCHL=0.28U, NCHVTO=0.140514, and PCHVTO=-0.0684537. The nominal delay value represents 36 ps.

For the simulation, we wrap the inverter netlist in a VHDL-AMS description to be able to apply the IS_NORMAL function.

Delays greater than 46 ps

We try to determine the probability *P* that the delay is greater than 46 ps. After 200 experiments importance sampling delivers $2 \cdot 10^{-2} < P = 3.24 \cdot 10^{-2} < 4.48 \cdot 10^{-2}$ and after 2000 standard Monte Carlo examples approximately $2.67 \cdot 10^{-2} < P = 3.45 \cdot 10^{-2} < 4.23 \cdot 10^{-2}$. After 200 standard Monte Carlo runs we obtain $1.23 \cdot 10^{-2} < P = 4 \cdot 10^{-2} < 6.77 \cdot 10^{-2}$. Thus, in this case, with 10 % of the Monte Carlo effort, importance sampling delivers the same accuracy.

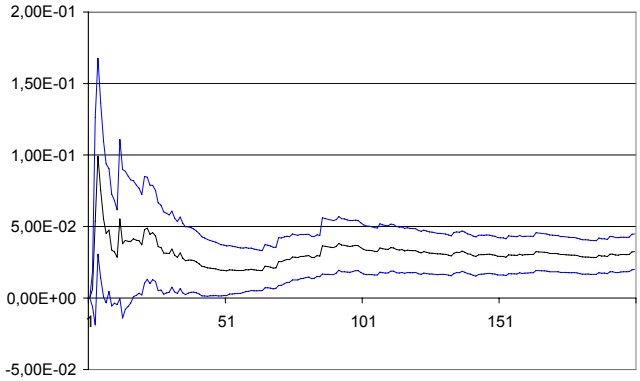

Figure 3. Estimated Prob(Delay > 46 ns) after 200 importance sampling runs

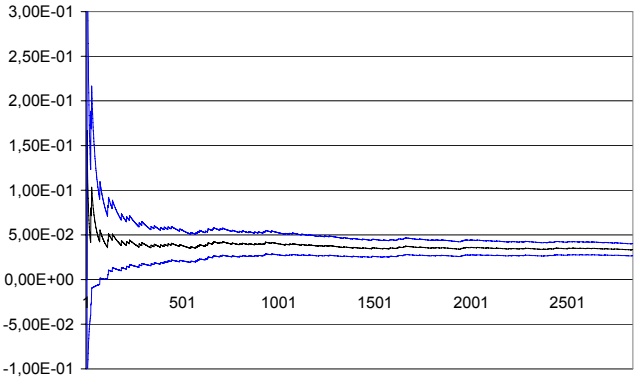

Figure 4. Estimated Prob(Delay > 46 ns) after 2000 Monte Carlo runs

Delays greater than 56 ps

We try to determine the probability P that the delay is greater than 56 ps. After 2000 importance sampling runs, we obtain $P = 5.70 \cdot 10^{-6} < 8.52 \cdot 10^{-6} < 11.35 \cdot 10^{-5}$. Using standard Monte Carlo simulation we would need about a million simulations to get an equivalent result. Figure 6 shows also a strong discontinuity after 91 runs. The discontinuity results in a combination of random parameter values with a relatively great weight. The effects of such discontinuities are reduced by increasing the number N of simulation runs. Then, a special weight has a smaller influence on the result (compare (5)).

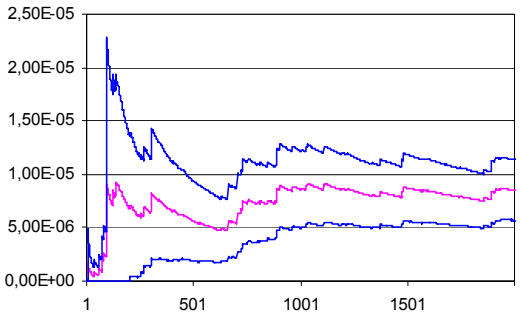

Figure 5. Estimated Prob(Delay > 56 ns) after 2000 importance sampling runs

D. *Multiplexer*

We determine the leakage current that flows into a multiplexer [10]. The nominal values of the considered transistor parameters are PCHWIDTH=10U, NCHWIDTH=3U, PCHL=0.13U, NCHL=0.13U, NCHVTO=0.140514, and PCHVTO=-0.0684537. (VDD=1.0). The current for nominal parameter values is 125 µA. We try to determine the Probability P that the current is greater than 200 µA.

After 2000 importance sampling runs we get the estimation $2.65 \cdot 10^{-11} < P = 3.31 \cdot 10^{-11} < 3.99 \cdot 10^{-11}$. Standard Monte Carlo simulation would require an unacceptably high number of runs.

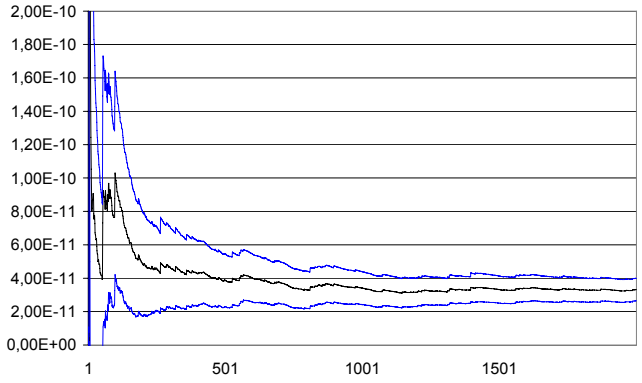

Figure 6. Estimated Prob(I > 200 µA) after 2000 importance sampling runs

V. CONCLUSION

Variance-reducing Monte Carlo techniques should or must be applied if very small probabilities have to be determined based on simulation results. Standard Monte Carlo simulation may require a huge number of simulation runs in such cases. Importance sampling is one alternative. This paper showed how the density function for importance sampling can be established based on the VHDL-AMS Statistical Analysis package. The introduced procedure for the IS_NORMAL function can easily be transferred to other density functions. The approach we presented offers the opportunity to apply importance sampling in existing VHDL/VHDL-AMS simulators with minor effort. Thus, an enabling technology for importance sampling is available.

Some simple examples demonstrated the application of the approach in order to analyze delay and power characteristics of basic IC blocks. The crucial task to determine a good density function for importance sampling was solved by exploiting the parameter sensitivity information. Further tests and improvements would be desirable to check whether this is a good way or not.

Another task is to check which other improvements of the standard Monte Carlo approach that are established in scientific computing [3, 4] are candidates for reducing the effort of statistical investigations in IC design.

ACKNOWLEDGEMENT

The authors thank especially Ernst Christen and David Smith for driving the activities developing the SAE J2748 standard. The work described in this paper was partly supported by the German Ministry of Education and Research (BMBF) within the project Sigma65 (ID 01M3080). The content is the sole responsibility of the authors.

REFERENCES

[1] Srivastava, A.; Sylvester, D.; Blaauw, D.: Statistical Analysis and Optimization for VLSI: Timing and Power. Springer, 2005.

[2] Kanj, R.; Joshi, R.; Nassif, S.: Mixture Importance Sampling and Its Application to the Analysis of SRAM Designs in the Presence of Rare Failure Events. Proc. DAC 2006, pp. 69-72.

[3] Robert, C.P.; Casella, G.: Monte Carlo Statistical Methods. Springer, 2004.

[4] Liu, J. S.: Monte Carlo Strategies in Scientific Computing. Springer, 2008 (softcover printing).

[5] Hesterberg, T. C.: Advances in importance sampling. PhD Dissertation, Statistics Departement, Stanford University, 1988.

[6] Denny, M.: Introduction to importance sampling in rare-event simulations. Eur. J. Phy 22 (2001) 403-411.

[7] Doorn, T. S.; ter Maten, E.J.W.; Croon, J.A.; Di Bucchianica, A.; Wittich, O.: Importance Sampling Monte Carlo simulations for accurate estimation of SRAM yield. Proc. ESSCIRC 2008, pp. 230-233.

[8] SAE J 2748 VHDL-AMS Statistical Packages. The SAE Electronic Design Automation Committee, Troy, MI, October 2006.

[9] Christen, E.; Bedrosian, D.; Haase, J.: Statistical Modeling with VHDL-AMS. Proc. FDL'07, pp. 44-49.

[10] 90nm-benchmark circuits SIMUCAD, Santa Clara, 30. August 2007. URL: http://www.simucad.com/news/2007_08_30_01.html

[11] Chiang, C; Kawa, J.: Design for Manufacturability and Yield for Nano-Scale CMOS. Springer, 2007.