# Radial Basis Functions Based Algorithms for Non-Gaussian Delay Propagation in Very Large Circuits

Dmytro Mishagli(✉) and Elena Blokhina

University College Dublin, Belfield, Dublin 4, Ireland
dmytro.mishagli@gmail.com, elena.blokhina@ucd.ie

**Abstract.** In this paper, we discuss methods for determining delay distributions in modern Very Large Scale Integration design. The delays have a non-Gaussian nature, which is a challenging task to solve and is a stumbling block for many approaches. The problem of finding delays in VLSI circuits is equivalent to a graph optimisation problem. We propose algorithms that aim at fast and very accurate calculations of statistical delay distributions. The speed of execution is achieved by utilising previously obtained analytical results for delay propagation through one logic gate. The accuracy is achieved by preserving the shapes of non-Gaussian delay distribution while traversing the graph of a circuit. The discussion on the methodology to handle non-Gaussian delay distributions is the core of the present study. The proposed algorithms are tested and compared with delay distributions obtained through Monte Carlo simulations, which is the standard verification procedure for this class of problems.

**Keywords:** Timing analysis · Statistical static timing analysis · Delay propagation · Uncertainty · Non-Gaussian · Graph optimisation

## 1 Introduction

The decrease of the feature size of modern Very Large Scale Integration (VLSI) design and the increase of the transistor count on a single chip inevitably approaches us to the end of Moore's law. Recently, Integrated Circuit (IC) technology has already reached the 5 nm technological node. At such scales, the role of uncertainty during the manufacturing process arises naturally due to physical fluctuations of various parameters such as the transistor channel width, its, length, etc. The design verification for a VLSI circuit has now become even more important since the complexity of design increases, which inevitably increase their cost. The standard way of circuit verification is to perform the timing analysis of a design [1,7,11,13]. The most reliable analysis is done by running Monte Carlo (MC) simulations that consider all possible variations of every parameter in a system. However, such computations can last for weeks for a single design.


V. V. Krzhizhanovskaya et al. (Eds.): ICCS 2020, LNCS 12141, pp. 217–229, 2020.
https://doi.org/10.1007/978-3-030-50426-7_17

Thus, semi-analytical methods based on delay models have been developed and are used in addition of MC simulations.

Deterministic Static Timing Analysis (or simply STA) can effectively take into account systematic process variations, and it was a dominant tool for several decades. However, STA gives too pessimistic predictions of delays, which significantly increases the cost of a chip in attempt to mitigate predicted delays. Therefore, the need for handling random correlated processes have arisen, and statistical approaches, known as Statistical Static Timing Analysis (SSTA), have been developed [2,5]. Within SSTA, all the delays in a system are treated as random variables (RVs) with some distributions. The goal of such an approach is to determine the mean value of the delay across selected paths and critical delays that may jeopardise the logic operation of the whole circuit. In addition, it is required to determine the standard deviation of the delay, its distribution, probability density function (PDF) and/or cumulative distribution function (CDF).

It is accepted now that the distribution of the delay generated by an individual logic gate is generally non-Gaussian [8]. A number of methods have been proposed to address this issue. However, treating non-Gaussian distributions of delays in a very large graph it still a challenge. Traditionally, approximations to the actual forms of distributions are used, as, for example, in studies [4,15,16], where the so-called *canonical model* of a delay has been proposed. Within this method, the delay is described as a linear function of parameter variations. Such approaches include (i) numerical approximations to the max-operator (see Sect. 3 for the details on this issue) and/or linearisation of nonlinear functions, and (ii) approximations to the actual distributions with Gaussians. For example, paper [12] discusses another modification of the canonical form based on adding a quadratic term and using skew-normal distributions.

In this study, we propose a fast and accurate algorithm for determining delay distributions taking into account their non-Gaussian nature. The problem of finding delays in VLSI circuits is formulated as that of a graph optimisation. The speed of the algorithm execution is achieved by utilising previously obtained analytical results for delay propagation through one logic gate, which was proposed in [6]. The accuracy is achieved by preserving the shapes of non-Gaussian delay distribution while traversing the graph of a circuit. This requires presenting a PDF of a gate's delay as a mixture of radial basis functions (RBFs) and solving the corresponding optimisation problem. The discussion on the methodology to handle non-Gaussian delay distributions is the core of the present study. The proposed optimisation strategies are incorporated in a traversal algorithm and tested and compared with delay distributions obtained through Monte Carlo simulations. The latter is the standard verification procedure for this class of problems.

The paper is organised as follows. In Sect. 2, we give a brief introduction to SSTA and discuss general statement of the problem. Section 3 discusses the model of delay propagation through a logic gate. The model allows us to built an algorithm for an accurate and fast calculation of the critical delay through a graph, which is summarised in Sect. 4. The key steps of the algorithm and

the optimisation problem are discussed in Sect. 5. Section 6 concludes this paper with the verification of the algorithm via simulations and overall discussion.

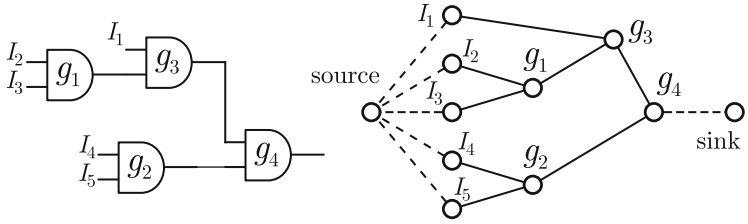


**Fig. 1.** Example logic circuit and its timing graph.

## 2 Statement of the Problem

### 2.1 Definitions

Throughout the paper, we will use the following commonly accepted terminology [2]. A logic circuit can be represented as a timing graph $G(E, N)$ as illustrated in Fig. 1 where the *graph* and its *paths* are defined as follows.

**Definition 1.** *A timing graph $G(E, N)$ is an acyclic directed graph, where $E$ and $N$ are the sets of edges and nodes correspondingly. Nodes reflect pins (logic gates) of a circuit. The timing graph always has only one source and one sink. Edges are characterised by weights $d_i$ that describe delays.*

The timing graph is called a *statistical timing graph* within SSTA when the edges of the graph are described by RVs. The task then is to determine the critical (longest) path.

**Definition 2.** *Let $p_i (i = 1, \ldots, N)$ be a path of ordered edges from the source to the sink in a timing graph $G$ and let $D_i$ be the path length of $p_i$. Then $D_{max} = \max(D_1, \ldots, D_n)$ is referred to as the SSTA problem of a circuit.*

Therefore, SSTA is aimed at determining the distribution of the circuit delay, which is equivalent to calculating the longest path in the graph formalism. Since SSTA operates with random variables, we will also use the following notation for the probability density function of the Gaussian distribution:

$$g(x|\mu, \sigma) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi}\sigma} \varphi\left(\frac{x-\mu}{\sigma}\right), \quad \varphi(x) \stackrel{\text{def}}{=} e^{-x^2/2} \tag{1}$$

where $\mu$ and $\sigma^2$ are the mean value and variance respectively, and $\varphi(x)$ is the Gaussian kernel function. The cumulative density function will be denoted as follows:

$$\Phi(x|\mu, \sigma) \stackrel{\text{def}}{=} \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{1}{\sqrt{2}}\frac{x-\mu}{\sigma}\right)\right], \tag{2}$$

where $\operatorname{erf}(x)$ is the error function.

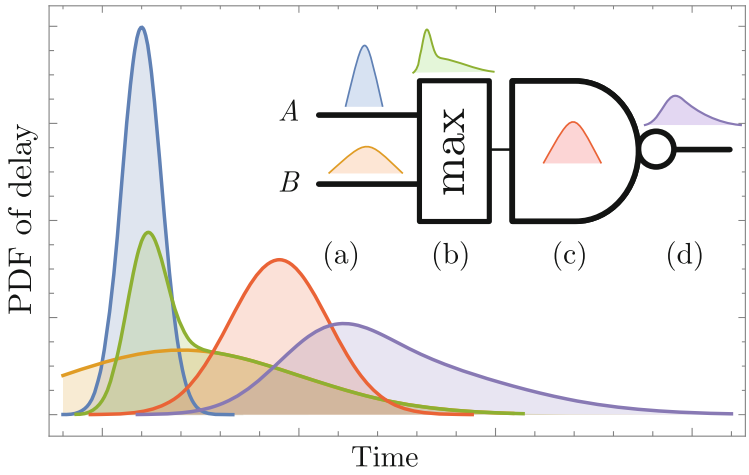


**Fig. 2.** Illustration of delay propagation through a logic gate. At stage (a), two signals arrive at the input of the gate. At stage (b), the max-operation is performed, which gives a skewed PDF. At the same time, the gate has its own operation time described by some distribution (c). Thus, the distribution of the gate delay (d) requires the convolution of the obtained distribution (b) and given (c). This convolution results in a new RV, which clearly has a non-Gaussian form.

### 2.2 Gate Level Analysis

We will start by explaining briefly how the overall delay is generated in a single logic gate. At the gate level, delay propagation is described by two operations: computing the maximum (max) of two delays entering a gate and the summation of the latter with the delay of the gate. From the statistical point of view, when these operations are applied to RVs, the delay of a gate with two inputs reads:

$$\max(X_1, X_2) + X_0, \tag{3}$$

where $X_1$ and $X_2$ are the RVs that describe the arrival times of input signals, and $X_0$ is the RV that gives the gate operation time. The operation of a logic gate in terms of the arrival and operation time distributions is presented in Fig. 2. Therefore, Eq. (3) is the *convolution* of the $\max(X_1, X_2,)$ and $X_0$ probability density functions. The analytical solution to combination (3) exists in a limited number of cases. We discuss these cases in the next section.

## 3 Model for a Logic Gate Delay

Consider a logic gate with two inputs, $A$ and $B$, and suppose that the gate operation time is distributed according to the normal law with a mean $\mu_0$ and a variance $\sigma_0^2$, i.e., it is a Gaussian RV. Assume now that the arrival times of both signals are also Gaussian RVs with means and variances $\mu_1$, $\sigma_1^2$ and $\mu_2$,

$\sigma_2^2$ respectively. Even if the individual distribution in the example of Fig. 2 are Gaussian, the application of formula (3) leads to a non-Gaussian distribution of the delay at the gate output as shown in that figure.

In our previous work [6], the exact expression for the PDF of such an RV was presented:

$$\mathfrak{f}(x) = \frac{1}{\sqrt{2\pi}} \sum_{\substack{i,j=1,2 \\ i \neq j}} \frac{1}{\tilde{\sigma}_i} \varphi\left(\frac{x - \mu_0 - \mu_i}{\tilde{\sigma}_i}\right) \Phi\left[\frac{1}{\sqrt{1 + \kappa_{ij}^2}} y(\ldots)\right], \tag{4}$$

where

$$\kappa_{ij} = \frac{\sigma_0 \sigma_i}{\sigma_j \tilde{\sigma}_i}, \quad \tilde{\sigma}_i = \sqrt{\sigma_0^2 + \sigma_i^2}, \quad y(x) = \frac{\sigma_i^2 (x - \mu_0) + \sigma_0^2 \mu_i}{\tilde{\sigma}_i^2 \sigma_j} - \frac{\mu_j}{\sigma_j}. \tag{5}$$

Expression (4) does not take into account possible correlations between the arrival signals. This issue will not be addressed in this study. Instead, we are interested in demonstrating how this exact solution can speed up SSTA for a given graph keeping precision high. Formula (4) assumes all initial delays (arrival and gate itself) to have Gaussian distributions. In principle, both the arrival signal and gate delay do not have to be Gaussian. If they can be decomposed into a linear superposition of Gaussian kernel functions, the PDF of the gate output delay can be presented as a linear combination of expressions (4) due to the linearity of the integration operation. This idea constitutes the core of a delay propagation algorithm which we discuss in the next sections.

## 4 Delay Propagation Algorithm

The algorithm for the calculation of the delay propagation through a timing graph is outlined below. The high-level description of the algorithm is shown in Fig. 3.

**Algorithm 1.** Returns a `list` with the parameters of Gaussian mixtures for each node of the graph $G$.

**Input:** graph $G$; distributions, means and variances for graph nodes

1. Perform preprocessing: decompose each non-Gaussian PDF for input nodes and gates and get corresponding Gaussian mixtures
2. Do forward propagation in $G$.
   **for** `node` **in** $G$:
   - calculate PDF $\mathfrak{f}(x)$, mean $\mu_{\text{gate}}$ and standard deviation $\sigma_{\text{gate}}$
   - represent via Gaussian mixture
   - append to a `list`

**Output:** a `list` with PDFs for all nodes in the RBF representation

This algorithm relies on the decomposition procedure. This procedure aims to represent a skewed (non-Gaussian) distribution with a mixture of RBFs that have Gaussian form, which allows one to use the result (4). In the next section, we discuss how such a decomposition can be performed.

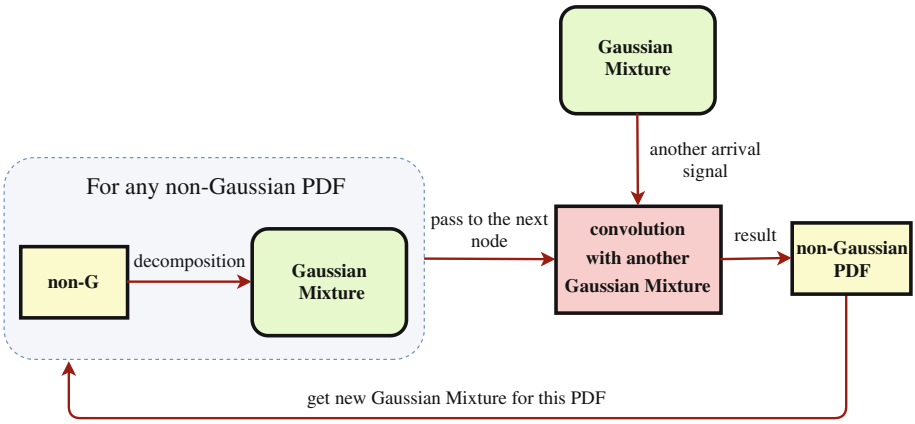


**Fig. 3.** High-level diagram of the Algorithm flow.

## 5 Optimisation

The exact function (4) can be written as $f_{\mathrm{rbf}}(x)$, a sum of RBFs; each of these RBFs has a Gaussian-like shape. In other words, it can be decomposed to a Gaussian mixture [9,14]. The decomposition procedure is equivalent to fitting the actual PDF with a sum of RBFs, which brings us to an optimisation problem. In this study, we discuss the minimisation of the sum of squares of the residuals

$$\min \sum_i |f_{\mathrm{rbf}}(x_i) - y_i|^2, \tag{6}$$

subject to constraints (specified below). Here $y_i$ are the data points corresponding to the actual function $\mathfrak{f}(x)$ that we want to fit.

Depending on the form of the RBFs, the minimisation of (6) can vary significantly, *e.g.*, an approximate function $f_{\mathrm{rbf}}(x)$ can be either linearly or non-linearly dependent on the fitting parameters. We discuss two alternative approaches below.

### 5.1 Choice of the Cost Function

Let us consider the RBFs $f_{\mathrm{rbf}}(x)$ in the following form:

$$f_{\mathrm{rbf}}(x) = \sum_{i=1}^{m} w_i \varphi\left(\frac{x - b_i}{c_i}\right), \quad \forall w_i, c_i > 0, \tag{7}$$

where $w_i$ are the weight coefficients, $b_i$ determine positions of the corresponding RBFs and $c_i$ are the shape parameters. The constraints on $w_i$ and $c_i$ are chosen so that the resulting mixture of RBFs has the meaning of a PDF. In the most general case, for $m$ RBFs there are $3\,\mathrm{m}$ parameters to be determined from the solution to the least-squares problem (6).

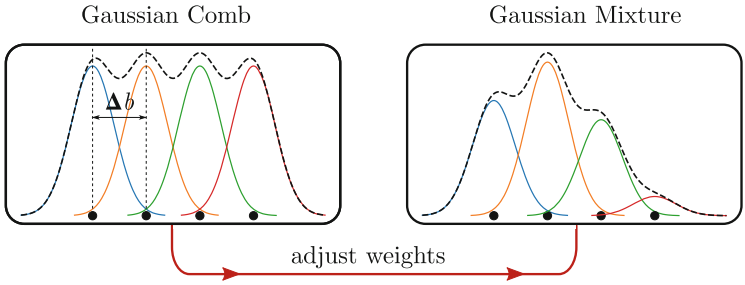


**Fig. 4.** A sketch of a Gaussian comb. For a given shape parameter $c$, the Gaussian kernels $\varphi(x)$, being equally separated by $\Delta b$, form a 1D grid, which we call a Gaussian comb. By adjusting the weights $w_i$, the Gaussian comb can give various shapes. The black dashed line shows the resulting curve from a mixture of Gaussian kernels in a comb.

Allowing all three parameters to vary, the fit function $f_{\mathrm{rbf}}(x)$ gives the desired shape using a small number $m$ of RBFs. At the same time, obtaining a minimum of (6) in such a case is not a trivial problem even for a few RBFs: the optimisation solver may stuck in one of the local minima, which makes this approach less reliable.

One can simplify the problem by setting a 1D grid of RBFs with given parameters $b_i$ and $c_i$, therefore, only the weights $w_i$ remain unknown. Note that we get only *linear* to the Gaussian kernels $\varphi(x)$ unknown parameters in such a case. To make the problem even simpler, let us fix the shape parameters $c_i$ for all kernels, $c_i = \mathrm{const}$, and let us distribute the kernels with equal separation $\Delta b$. We shall call such mixtures of RBFs a *Gaussian comb* (see Fig. 4). At the same time, such a simplification will require more terms in a mixture of RBFs to get the desired shape.

Thus, to construct the Gaussian comb, one needs to set up the number $m$ of RBFs in the comb, the Gaussian comb step $\Delta b$ and the shape parameter $c$. The weight coefficients $w_i$ then to be determined from (6). In this study, we assume $m$ and $c$ are known *a priori*, and the step $\Delta b$ is determined as

$$\Delta b = \frac{\Delta y}{m}, \tag{8}$$

where $\Delta y$ is the effective interval of values over which the function (PDF) must be fit (we call it *bandwidth*). The choice of the bandwidth is discussed below. In principle, the optimisation problem can be formulated in such a way that $m$ and $c$ are determined simultaneously with the weights $w_i$, making the solution self-consistent. This will be reported elsewhere.

The distinguishable feature of such RBF decompositions is that one obtains simple functions that allow fast and simple computations of a mean and a vari-

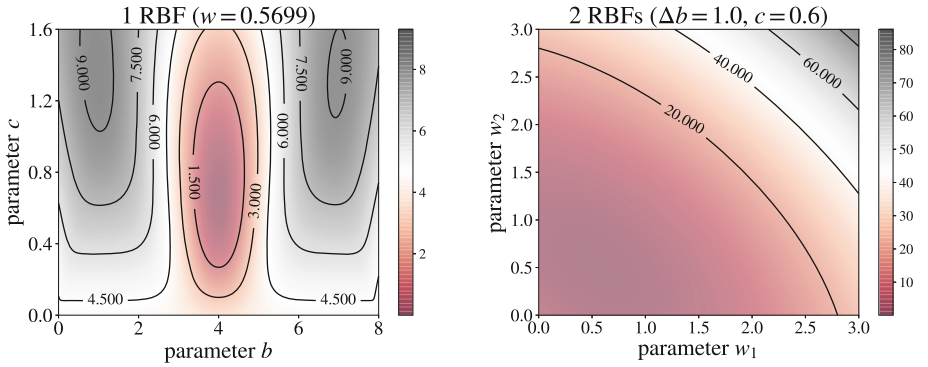


**Fig. 5.** Cost functions for the least square problem of fitting a Gaussian PDF with $\mu = 4$ and $\sigma = 0.7$. *Left:* only one RBF is used; for $w = 0.5699$, the minimum is located at $b = 4$ and $c = 0.7$. *Right:* two RBFs are used; for $b_1 = 3.5$, $b_2 = 4.5$, $\Delta = 1.0$ and $c = 0.6$, the minimum is located at $w_1 = w_2 = 0.5299$.

ance for the PDF:

$$\mu_{\text{rbf}} = \sqrt{2\pi} \sum_{i=1}^{m} w_i b_i c_i, \quad \sigma^2_{\text{rbf}} = \sqrt{2\pi} \sum_{i=1}^{m} w_i c_i (b_i^2 + c_i^2) - \mu^2_{\text{rbf}}; \tag{9}$$

for a case of Gaussian comb, $\forall c_i = c$.

### 5.2 Comparison

For the sake of illustration, consider a problem of fitting the Gaussian PDF, $g(x|\mu = 4, \sigma = 0.7)$, with only one RBF. If we pre-set the weight $w$, only two parameters, $b$ and $c$, are left to find. Choosing $w = 1/(\sqrt{2\pi}0.7) \approx 0.6599$, the parameters should be then exactly $b = 4$ and $c = 0.7$. A contour plot of the cost function (6) for such a case is shown in Fig. 5 on the left. The cost function has only one minimum, which lies in a valley (the point with $b = 4$ and $c = 0.7$). However, if we allow the weight $w$ to vary, there will be 3 parameters to find and local minima appear. Increasing the number $m$ of RBFs inevitably brings us to a problem of omitting such minima.

The Gaussian comb consist of two Gaussian kernels with $c = 0.6$ and $\Delta b = 1$ ($b_1 = 3.5$ and $b_2 = 4.5$) leads to the cost function shown in Fig. 5 on the right. One can see that the minimum is located in the bottom of a steep well that rises dramatically as one goes away from the minimum. Thus, there will be no difficulties in finding the minimum in higher dimensions, moreover, this problem allows exact solution.

Consider another example. The PDF $f_{\text{LN}}(x)$ of the lognormal distribution reads

$$f_{\text{LN}}(x) = \frac{1}{x} \cdot \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln x - \mu)^2}{2\sigma^2}\right), \tag{10}$$

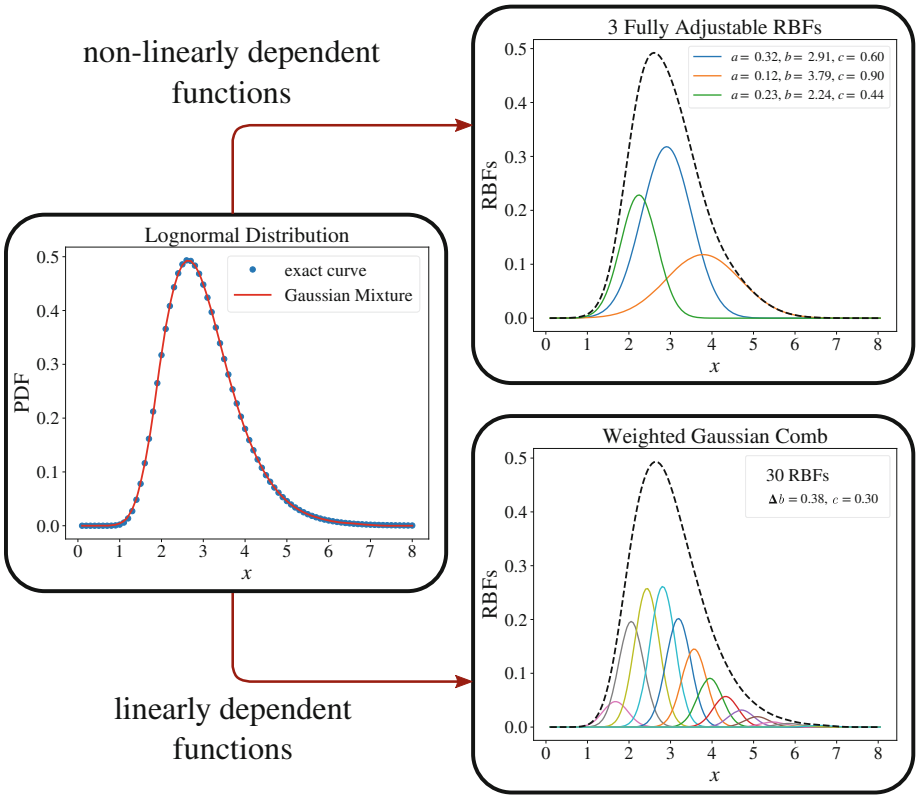


**Fig. 6.** Case study: representation of the lognormal distribution with a Gaussian mixture. Two strategies are realised: (i) 3 RBFs with nonlinear to kernels fitting parameters and (ii) 30 RBFs forming the Gaussian comb with linear to kernels parameters. Both strategies give the perfect fit and desired values of $\mu_{\mathrm{LN}} = 3$ and $\sigma_{\mathrm{LN}} = 0.9$ via the approximation (9).

where $\mu$ and $\sigma^2$ are the mean and variance of the corresponding normal distribution. The reciprocal relation between these $\mu$ and $\sigma^2$ and the mean and variance of the lognormal distribution, $\mu_{\mathrm{LN}}$ and $\sigma^2_{\mathrm{LN}}$, is as follows:

$$\mu_{\mathrm{LN}} = \exp\left(\mu + \frac{\sigma^2}{2}\right), \quad \sigma^2_{\mathrm{LN}} = \left[\exp\left(\sigma^2\right) - 1\right] \cdot \exp\left(2\mu + \sigma^2\right). \tag{11}$$

Two different decompositions of (10) with $\mu_{\mathrm{LN}} = 3$ and $\sigma_{\mathrm{LN}} = 0.9$ are shown in Fig. 6. One can see that both approaches give perfect result, while require different number of RBFs: (i) 3 RBFs with 3 fitting parameters each ($w_i$, $b_i$ and $c_i$) and (ii) 30 RBFs in the Gaussian comb with 30 fitting parameters (weights $w_i$). The bandwidth is chosen as $\Delta y = [\mu_{\mathrm{LN}} - 4\sigma_{\mathrm{LN}}, \mu_{\mathrm{LN}} + 4\sigma_{\mathrm{LN}}]$. We shall keep this choice for the bandwidth in this study although it is not optimal

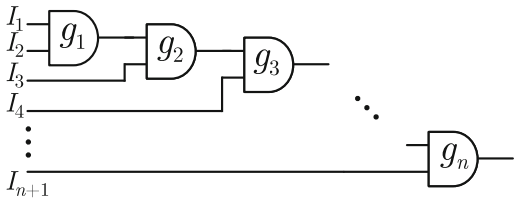


**Fig. 7.** A sequence of logic gates used in simulations.

(distributions can be significantly skewed). This issue will be addressed in a separate study.

In the next section, the proposed Algorithm with the decomposition strategies discussed above is tested and compared with the numerical experiments (Monte Carlo simulations). The goal is to proof the concept, determine possible issues and define future steps in the research.

## 6 Verification and Discussion

Let us investigate whether an error in computing the delays' PDF is accumulating or not when the decomposition into Gaussian kernels is used. To do so, we consider a model circuit shown in Fig. 7. The inputs' delays $I_i$ $(i = 1, \ldots, n)$ are described by the corresponding RVs, $X_i$. For simplicity (and without any loss of generality), the operation time of the gates is considered to be the same, thus, described by an RV $X_0$. From the mathematical point of view, the forward traversing of such a sequence of gates is equivalent to computing the chained expressions of type (3). Thus, for the $n$th gate we have

$$\max\{\underbrace{\ldots \max[\max(X_1, X_2) + X_0, X_3] + X_0 \ldots,}_{n-1 \text{ times}} X_{n+1}\} + X_0. \tag{12}$$

We have conducted a series of runs of the Algorithm and MC simulations for the sequence of $n = 20$ (the source code is available from [10]). We have chosen the initial delays $X_i$ to be distributed as $X_i \sim \mathcal{N}(\mu_i, \sigma_i)$. The values for the inputs' means and standard deviations, $\mu_i$ and $\sigma_i$, were randomly drawn from $\sim \mathcal{U}(2, 7)$ and $\sim \mathcal{U}(0.2, 1.3)$ respectively for each run. One of the realisations of the experiments is shown in Fig. 8. Since the absolute values of delays are not important in the present study, the performance of the algorithm is measured by *relative errors* in the mean values and standard deviations of delays with respect to the Monte Carlo simulations.

For the Gaussian comb, $m = 55$ kernels have used with the shape parameter $c = 0.15$. The relative error is less than 0.01% and remains at that level until it starts to grow dramatically (see Fig. 8). This occurs when the chosen topology of the comb becomes non-optimal, as it is shown for node 19. Also note that the relative error in the standard deviation increased faster than that for the mean, which is expected.

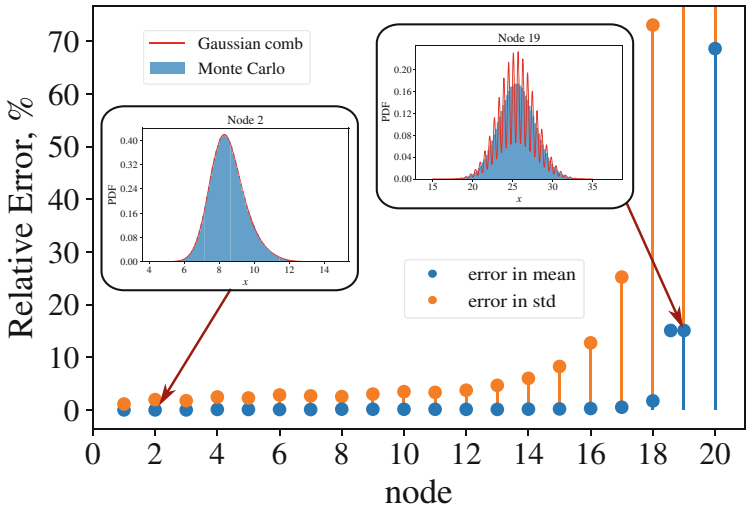


**Fig. 8.** Results for the Gaussian comb approach ($m = 55$ kernels with $c = 0.15$): the relative error in determination of the mean and standard deviation with respect to the Monte Carlo values versus number of gates passed in the sequence. When the bandwidth $\Delta y$ becomes large, the topology of the Gaussian comb is no longer optimal and the comb sprawls (node 19). For the details of the simulations see the text.

For the case of 3 RBFs with non-linear fitting parameters, the algorithm has given poor performance. As is discussed in the previous section, the optimisation problem (6) in this case is sensitive to small deviations in an initial guess or change of the bandwidth $\Delta y$. Thus, it has not been possible to finish the traversal of the graph successfully (a Trust Region Algorithm [3] for the constrained optimisation was used), and the results for this approach are not presented.

The obtained results allow us to conclude that the problem of VLSI circuit delay indeed can be solved using (i) exact solution for a single gate's delay PDF and (ii) decomposition of non-Gaussian functions into Gaussian mixtures. The detailed discussion and conclusions are as follows.

(i) The exact formula for an output logic gate delay, the convolution of $\max(X_1, X_2)$ and $X_0$ for Gaussian RVs $X_i$ $(i = 0, 1, 2)$, allows one to build a closed-loop algorithm for forward traversal of a delay through a timing graph $G$. The requirement for this is that non-Gaussian PDFs of delays are presented via Gaussian mixtures, sums of RBFs of Gaussian form. The decomposition is equivalent to solving the minimisation problem (6). We have considered two different strategies for this problem.

(ii) Within the first strategy, for $m$ RBFs it is required to determine $3m$ parameters, $3m - 1$ of which are coefficients in the *arguments* of the RBFs. The advantage is that only a few of RBFs is enough to obtain a fit with a desired accuracy, but the drawback is sensitivity to small changes in the parameters such as initial guess, choice of bandwidth $\Delta y$, etc.

(iii) The second strategy relies on a pre-set grid of equally separated Gaussian kernels $\varphi(x)$ with the same shape parameter, a *Gaussian comb*. In such a case, only coefficients that are linear multipliers to RBFs should be determined. The obvious advantage of this approach is that the optimisation problem allows the exact solution, however the required number $m$ of RBFs increases dramatically, which increases (linearly to $m$) computation costs.

(iv) The comparison with MC simulations shown in Fig. 8 proofs the concept: the timing graph can be forward traversed with the relative error less than 0.01% by decomposing real PDFs into corresponding Gaussian mixtures at each node. However, when the topology of the Gaussian comb is fixed, it leads to sprawling of the latter as the bandwidth $\Delta y$ becomes large.

(v) In principle, the optimisation problem of finding the weights $w_i$ for the Gaussian comb can be solved together with the problem of finding optimal number $m_*$ and shape parameter $c_*$ of kernels in the comb. This should not only prevent the comb sprawling but also speed up the graph traversal procedure noticeably. At the same time, the optimisation problem for the 3-RBF case should be analysed rigorously to avoid slipping in local minima. This can be an alternative to the Gaussian comb decomposition. These issues will be addressed in a separate study.

**Acknowledgement.** This work has emanated from research supported in part by Synopsys, Ireland, and a research grant from Science Foundation Ireland (SFI) and is co-funded under the European Regional Development Fund under Grant Number 13/RC/2077.

## References

1. Bhasker, J., Chadha, R.: Static Timing Analysis for Nanometer Designs. A Practical Approach. Springer, Heidelberg (2009). https://doi.org/10.1007/978-0-387-93820-2
2. Blaauw, D., Chopra, K., Srivastava, A., Scheffer, L.: Statistical timing anlaysis: from basic principles to state of the art. IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst. **4**(8), 589–607 (2008)
3. Byrd, R.H., Schnabel, R.B., Shultz, G.A.: A trust region algorithm for nonlinearly constrained optimization. SIAM J. Numer. Anal. **24**(5), 1152–1170 (1987). https://doi.org/10.1137/0724076
4. Chang, H., Zolotov, V., Narayan, S., Visweswariah, C.: Parameterized block-based statistical timing analysis with non-Gaussian parameters, nonlinear delay functions. In: Proceedings of DAC, pp. 71–76, June 2005. https://doi.org/10.1145/1065579.1065604
5. Forzan, C., Pandini, D.: Statistical static timing analysis: a survey. Integr. VLSI J. **42**(3), 409–435 (2009). https://doi.org/10.1016/j.vlsi.2008.10.002, special Section on DCIS2006
6. Freeley, J., Mishagli, D., Brazil, T., Blokhina, E.: Statistical simulations of delay propagation in large scale circuits using graph traversal and kernel function decomposition. In: Proceedings of SMACD, July 2018
7. Gerez, S.H. (ed.): Algorithms for VLSI Design Automation. Wiley, Hoboken (1998)

8. Lavagno, L., Markov, I.L., Martin, G., Scheffer, L.K. (eds.): Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology. CRC Press, Boca Raton (2016)
9. McLachlan, G., Peel, D.: Finite Mixture Models. Wiley Series in Probability and Mathematical Statistics. Wiley, Hoboken (2000)
10. Mishagli, D.: RBF approximation for non-Gaussian delay propagation in VLSI: Code (2020). https://doi.org/10.5281/zenodo.3749750
11. Orshansky, M., Nassif, S., Boning, D.: Design for Manufacturability and Statistical Design: A Constructive Approach. Series on Integrated Circuits and Systems. Springer, Heidelberg (2008). https://doi.org/10.1007/978-0-387-69011-7
12. Ramprasath, S., Vijaykumar, M., Vasudevan, V.: A skew-normal canonical model for statistical static timing analysis. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **24**(6), 2359–2368 (2016). https://doi.org/10.1109/TVLSI.2015.2501370
13. Sapatnekar, S.: Timing. Springer, Heidelberg (2004). https://doi.org/10.1007/b117318
14. Titterington, D., Smith, A., Makov, U.: Statistical Analysis of Finite Mixture Distributions. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. Wiley, Hoboken (1985)
15. Visweswariah, C., Ravindran, K., Kalafala, K., Walker, S.G., Narayan, S.: First-order incremental block-based statistical timing analysis, pp. 331–336 (2004). https://doi.org/10.1145/996566.996663
16. Visweswariah, C., et al.: First-order incremental block-based statistical timing analysis. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 25(10), 2170–2180 (2006). https://doi.org/10.1109/TCAD.2005.862751