Statistical Simulations of Delay Propagation in Large Scale Circuits Using Graph Traversal and Kernel Function Decomposition

Jennifer Freeley, Dmytro Mishagli, Tom Brazil and Elena Blokhina
School of Electrical and Electronic Engineering, University College Dublin, Ireland

***Abstract*—In this paper we propose a new methodology to determine the delay of combinational logic circuits within the framework of statistical static timing analysis (SSTA). A new algorithm for the traversing of the timing graph is created and combined with a new technique of kernel function decomposition to find delay propagation through such a circuit. Assuming initial delays of the input signals and operation time of gates to be normally distributed, the exact analytical solution for a non-Gaussian probability density functions (PDF) of the resulting delay is obtained. Then, the approximation of a non-Gaussian PDF by a linear combination of kernel functions is proposed, and the initial Gaussian assumption is relaxed. This allowed us to build a novel closed-loop algorithm for the calculation of delay propagation in combinational circuits. Possible extensions and future steps are discussed.**

I. Introduction

The continuous reduction of feature size is creating new challenges for the timing analysis of digital integrated circuits (ICs) as process-related uncertainties begin to dominate behavior. A digital IC design must operate safely at the specified frequency of the clocks without any timing violations, which is typically checked by timing analysis tools [1], [2].

As technology continues to scale down, the impact of process variations (such as process-voltage-temperature variations) on timing grows (see e.g. [3], [4]). Also variations arise from the manufacturing process (e.g. transistor length is difficult to control exactly). Moreover, with decreasing size of transistors and interconnect width, the variations of electrical characteristics can be of the same order as nominal values. This has created a new wave of interest in timing verification and introduced the need for an update to traditional algorithms, which has led to the use of Monte Carlo (MC) simulations. MC analysis becomes computationally expensive for large ICs. As a result, the use of approaches that are a counterpart of Monte Carlo known as statistical static timing analysis (SSTA) is increasing, with Monte Carlo analysis acting as a 'golden' reference.

Within the SSTA [5], all delays are treated as random variables with corresponding probability density functions (PDFs). As we will show in the next section of this paper, three main challenges of SSTA have been addressed with different degrees of success: (i) impact of spatial correlations, (ii) non-analytical operations such as $\max$, and (iii) non-Gaussian distributions of variations.

The $\max$ operation significantly distorts the distribution of the delay. Even though the actual gate and interconnect delays are considered to be Gaussian, when delay propagates it quickly becomes non-Gaussian. Certain combinations of gate distributions can result in the accumulation of a considerable error. The effect may not be critical for some parameters but can lead to significant errors in others.

978–1–5386–5153–7/18/$31.00 © 2018 European Union

Recent papers show an increasing interest in SSTA [6]–[9], and the need for fast algorithms for stochastic analysis has increased. For this reason, we first propose to initiate a new statistical approach that relies on an analytical treatment of one gate. We develop a closed form expression for the probability density function of the "maximum" operation ($\max$) with a delay representing a gate and/or interconnect delay. The proposed algorithm can be extended to a number of practically important cases such as correlated variables and multiple inputs. As is expected, working with an analytical approach would not be time and resource demanding and would allow a faster analysis of a very large circuit.

Before completing timing analysis, it is necessary to traverse the timing graph (see section II). Hence we propose an algorithm that involves the unique analysis of a matrix containing all circuit paths, which reveals an optimal sequence of sum and maximum operations to be performed on gate delays so that the delay of all nodes in all graph paths can be computed without the need for backtracking.

The aim of this paper is to provide updated algorithms for traversing the timing graph as part of SSTA and for the statistical simulations performed to compute the circuit delay. The paper is divided as follows: Section II presents a general statement of the problem as the delay propagation through a graph of a combinational logic circuit. Section III details the algorithm for traversing the graph and obtaining the optimal sequence of maximum and sum operations to perform. Section IV describes the steps involved in the analysis of delay propagation through one graph node and provides a novel technique to calculate gate delays. Section IV presents the performance of the algorithm proposed as well as discussions on further development of the method and conclusions.

II. Statement of the Problem

A combinational logic circuit traditionally is represented by its timing graph. Edges of the graph represent gates and vertex set is for inputs and outputs of the logic gates (see Fig. 1). Since the gates have internal structure presented by corresponding combination of transistors, this results in a characteristic time needed for gates to operate. This is one of the sources of delays in a circuit. Due to delays, input signals can have different arrival times and, therefore, the delay of a gate is determined by the maximum of input delays.

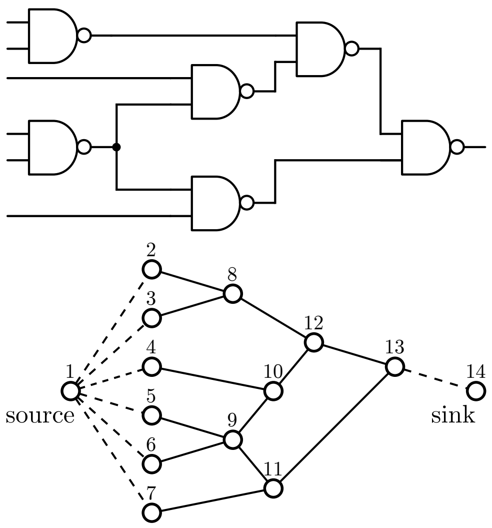

Fig. 1. An example combinational circuit and its timing graph.

Thus, the main problem of timing analysis for such circuits can be formulated as the mathematical problem of calculating the $\max$ function of arrival times. In other words, this is a problem for the graph optimisation.

In addition to the arrival times, the operation time of a gate can have a significant impact on circuit delay. In such a case the delay of a gate itself should be added to the result of the $\max$ function. The situation is straightforward in the case of deterministic timing analysis, but it is not the case when uncertainty arises. When it is necessary to include variations of parameters, SSTA is required. In such a case arrival and gate operation times are described by random variables (RVs) given by corresponding distributions.

The situation becomes even more dramatic in a lower scale, when variations are of the order of the nominal values of parameters. Here, the problem of calculating of the $\max$ of two or more RVs discussed in the Introduction occurs. At the same time, it is impossible to ignore interconnect delays. However, the latter can be added to the gate delay. Thus, the procedure of the delay computation in one gate with two inputs can be written as

$$\ldots + \max(\tau_i, \tau_j) + \tau_{\text{g}} + \tau_{\text{int}} + \ldots, \tag{1}$$

where τ_i, τ_j are input signal delays, τ_{g} is a gate delay and τ_{int} is an interconnect delay. In the case of Gaussian distributions, the convolution for the latter two terms can be easily performed, resulting to another Gaussian distribution. For simplicity, the interconnect delay is not included in the traversal algorithm, but can be easily introduced.

Here the problem of backtracking appears [10], since the expressions like (1) depend on the previously obtained values of arrival times. In the next section, we present our algorithm that allows us to find the gate-to-gate sequence of graph traversal, removing the need for backtracking.

III. Improved Graph Traversal Algorithm

Graphs are typically described by corresponding adjacency matrices. In order to build the algorithm for a graph traversing, we assume that (i) all adjacency matrices to be analysed are upper triangular, with zeros on the diagonal, and (ii) there are no isolated paths. The initial step is to find all simple paths (ones not containing two or more occurrences of the same node [10]), arrange them in descending order of length, and display in a matrix form. For the sake of example, the path matrix $\mathbf{P}$ for the graph from Fig. 1 is shown in Fig. 2, but it should be noted that the construction of the matrix is simple so the approach is scalable to larger circuits.

$$\mathsf{P} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 9 & 10 & 0 & 12 & 13 & 14 \\
1 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 9 & 10 & 0 & 12 & 13 & 14 \\
1 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 9 & 0 & 11 & 0 & 13 & 14 \\
1 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 9 & 0 & 11 & 0 & 13 & 14 \\
1 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 12 & 13 & 14 \\
1 & 0 & 3 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 12 & 13 & 14 \\
1 & 2 & 0 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 12 & 13 & 14 \\
1 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 11 & 0 & 13 & 14
\end{bmatrix}$$

Fig. 2. Path matrix for the graph in Fig. 1.

Each node has several attributes: τ_{g} is the delay due to the operation time of the gate that the node n represents, which is given; τ_n represents the node delay, which we compute according to Algorithm 1. The list of upstream nodes for each node includes all of the nodes that appear to the left of n in each row of the path matrix $\mathbf{P}$ that n appears in. For example, nodes 1, 2 and 3 are the upstream nodes for node 8 in the matrix in Fig. 2. The list of input nodes to each node is found from the adjacency matrix and refers to nodes that are adjacent to n and are the alternate endpoints of the edges that are incident to it.

Algorithm 1: SequenceAlgorithm returns an expression for the delay of all nodes in the path matrix $\mathbf{P}$, given the node structure with float numbers τ_n, τ_{g}, and lists inputs and upstreams

```
Function checkMax(n):
    for node ∈ n.upstreams do
        if node ∉ n.inputs then
            remove node from n.upstreams
    if n.inputs ∈ n.upstreams then
        $n.\tau_n \leftarrow n.\tau_{\text{g}} + \max(n.inputs.\tau_n)$
Function checkSum(n):
    if n.inputs ∈ n.upstreams then
        if length(n.inputs) = 1 then
            $n.\tau_n \leftarrow n.\tau_g + n.inputs.\tau_n$
foreach $n \in \mathbf{P}$ do
1   n.upstreams ← getUpstreamNodes($n, \mathbf{P}$)
2   checkSum($n$)
3   if $n \notin$ sumNodes then
        checkMax($n$)
4 return $n.\tau_n$ for $n \in \mathbf{P}$
```

All nodes that appear in the path matrix $\mathbf{P}$ excluding the source node are added to a list that is subsequently ordered in terms of node number. For each node n, the first step is to find all of its upstream nodes. The method of finding these nodes is not explained in detail here, as it is straightforward, but is shown by the function call `getUpstreamNodes`$(n, \mathbf{P})$ that takes the path matrix and node to be processed as arguments. The list of upstream nodes is then passed to the `checkSum` procedure.

If the node n has only one input node, the sum operation is required. For this reason, the `checkSum` function confirms that the list of upstream nodes contains the input node to node n and that the list of input nodes has length 1. If this is the case, the delay of node n (i.e. $n.\tau_n$) will be equal to the sum of the delay of the gate that node n represents and the delay of its input node. In the path matrix presented here, the delay of node 2 is computed using the sum operation from $\tau_2 = \tau_1 + \tau_{\text{g},2}$.

If there is more than one input to a gate, the total input delay is equal to the maximum of the arrival times of its inputs. In terms of the timing graph, these arrival times are given by the delays of the input nodes to node n. The `checkMax` procedure first removes nodes from the list of upstream nodes that are not input nodes to node n. It then checks that the remaining list contains all of the input nodes to node n. If this is the case, $n.\tau_{\text{g}}$ is added to the maximum of the delays of the input nodes to n to give $n.\tau_n$. Node 9 offers an example of the max operation where $\tau_9 = \max(\tau_5, \tau_6) + \tau_{\text{g},9}$.

As a result, we obtain an *explicit sequence* of operations required to obtain analytically the expression of total delay between any two nodes. The graph traversal algorithm allows us to see the structure of the graph and its underlying correlations, which is a significant advantages in the analysis of circuit correlations. In the next section we consider the delay propagation inside logic gates.

IV. Logic gate delay

As soon as the sequence of operations required to find the delay between two nodes is found from the Algorithm presented in Section III, one must use the gate delays $\tau_{\text{g},i}$ involved in this sequence to calculate the actual delay. Here we present a new technique of delay calculation through a gate whose main advantage is the ability to *handle non-Gaussian distributions* resulting from the max operation.

Let us consider a simple logic gate with two inputs, and let us suppose we have two arrival signals with known delays, A and B, and these delays are distributed normally. We shall describe these delays as Gaussian variables, X_1 and X_2, with given mean values, μ_1 and μ_2, and variances, σ_1^2 and σ_2^2.

The computing of a maximum is not a straightforward task since the operation is non-linear (see e.g. [11]). The situation is even more complicated when applied to randomly distributed variables. For the independent RVs X_1 and X_2 with probability density functions (PDF) $f_1(x)$ and $f_2(x)$ and cumulative distribution functions (CDF) $\Phi_1(x)$ and $\Phi_2(x)$ the PDF of another RV, $\max(X_1, X_2)$ is given by a well-known formula [12], which is valid for any type of distribution: $f_{\max}(x) = f_1(x)\Phi_2(x) + f_2(x)\Phi_1(x)$.

We are not considering correlations between variations in this study (this will be reported elsewhere). However, we would like to point out that introducing correlations at this stage will not make any substantial difficulties, and all obtained results can be easily generalized to the correlated case (see e.g. [13]).

Let us now assume that the operation time of a gate leads to a delay that can be described as a Gaussian RV X_3 with known mean μ_3 and variance σ_3^2, and the PDF $f_3(x)$. Then, the total delay of a gate is $X = \max(X_1, X_2) + X_3$. In terms of distributions, the convolution of $\max(X_1, X_2)$ and X_3 should be performed:

$$(f_{\max} * f_3)(x) = \int_{-\infty}^{\infty} f_{\max}(x') f_3(x - x') dx'. \tag{2}$$

For the case of Gaussian functions $f_1(x)$, $f_2(x)$ and $f_3(x)$, the latter integral can be computed analytically. Therefore, after the corresponding algebra one obtains the PDF of the total gate delay $f_{\text{tot}}(\ldots)$ as a function of all initial delays in the following form:

$$f_{\text{tot}}(x; \mu_1, \sigma_1, \mu_2, \sigma_2, \mu_3, \sigma_3) = \frac{1}{2\sqrt{2\pi}}[I_{12}(x) + I_{21}(x)], \tag{3}$$

here $I_{ij}(x)$ $(i \neq j = 1, 2)$ are

$$I_{ij}(x) = \frac{1}{\sqrt{\sigma_3^2 + \sigma_i^2}} \exp(-\alpha_i x^2 + \beta_i x + \gamma_i) \times \left\{ 1 + \operatorname{erf}\left[b_{ij}(x) \left(\frac{1}{1 + a_{ij}^2} \right)^{\frac{1}{2}} \right] \right\}, \tag{4}$$

where

$$\begin{aligned} \alpha_i &= \frac{1}{2} \frac{1}{\sigma_3^2 + \sigma_i^2}, \quad \beta_i = 2(\mu_3 + \mu_i)\alpha_i, \\ \gamma_i &= -(\mu_3 + \mu_i)^2 \alpha_i = -\frac{1}{2}(\mu_3 + \mu_i)\beta_i \\ a_{ij} &= \frac{\sigma_3 \sigma_i}{\sigma_j \sqrt{\sigma_3^2 + \sigma_i^2}}, \\ b_{ij}(x) &= \frac{\sigma_i^2 x + \sigma_3^2 \mu_i - \sigma_i^2 \mu_3 - (\sigma_3^2 + \sigma_i^2)\mu_j}{\sqrt{2}(\sigma_3^2 + \sigma_i^2)\sigma_j}. \end{aligned} \tag{5}$$

It is clearly seen that the resulting delay has non-Gaussian form. For the purpose of a closed-loop algorithm, the simplest solution would be to approximate at each step the distribution (2) with a Gaussian by matching two first moments [14]. However, as it was also indicated in the Introduction, this leads to huge deviations of the result from actual distributions. In order to keep both non-Gaussian form of delays and analyticity of the solution presented above, we propose to approximate non-Gaussian PDFs with the following linear combination of kernel functions:

$$f_{\text{approx}}(x) = \sum_{i=1}^{n} a_i \exp\left(-\frac{(x - b_i)^2}{2c_i^2} \right), \tag{6}$$

where we determine the unknown coefficients a_i, b_i and c_i by solving a corresponding optimisation problem subject to constraints $\sum_i a_i c_i = \frac{1}{\sqrt{2\pi}}$ and a_i, b_i, c_i to be > 0 in order the new function $f_{\text{approx}}(x)$ has meaning of PDF.

This allows us to build the Algorithm 2.

Algorithm 2: GATEDELAY finds the PDF of a delay of combinational circuits.

```
Input: $\mu_i$, $\sigma_i$ of gates' delays, number of gates $N_{\mathrm{G}}$
Output: PDF, mean and variance of the delay of a circuit
1 for $i \leftarrow 1, N_{\mathrm{G}}$ do
2     Compute the PDF of a total gate delay (3)–(5)
3     Perform optimization and find all $a_i$, $b_i$, $c_i$
4     Approximate actual PDF with (6)
5 return final PDF of a circuit
```

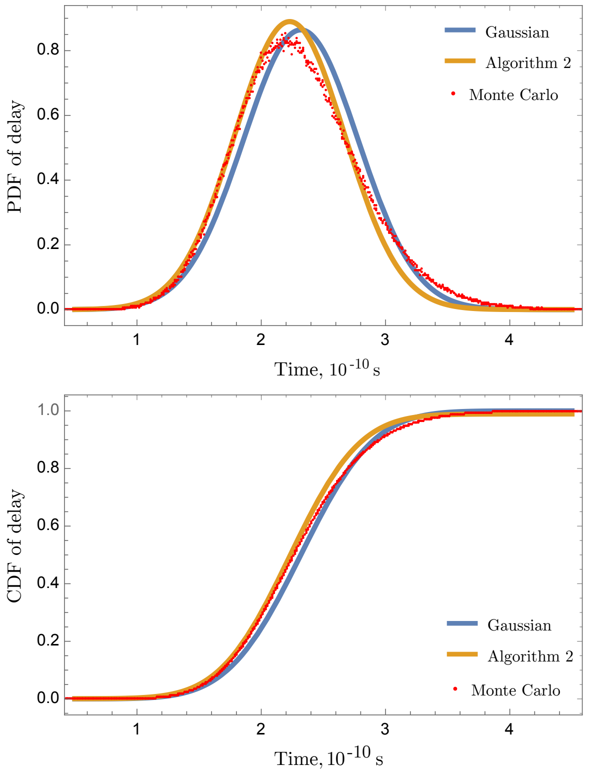

Fig. 3. Example of the Algorithm 2 performance versus MC. The sum of three kernel functions (6) was used. Additionally, the standard Gaussian approximation is shown for the sake of comparison.

V. Verification and Discussion

To test the performance of the Algorithm 2 a set of realistic parameters of input signal delays for the circuit from Fig. 1 were chosen. All input delays, as well as the gate delay, were chosen normally distributed. The gate delay was assumed equally distributed for all gates. The algorithm and MC were run for a combination of 6, 42 and 258 gates. An example of the resulting PDF and CDF is presented in Fig. 3, in addition the results for pure Gaussian approximation are shown. The proposed Algorithm 2 was in average $\sim 10^2$ times faster than the MC simulation with 10^7 samples.

As it is seen from Fig. 3, the proposed approximation (6) gives more accurate results compared to Gaussian approximation, since such an approach allows one to take into account skewness of non-Gaussian distributions. At the same time, deviations from MC can be explained by inaccuracy of the optimisation performed (a high-level programming language was used to perform the fit), and a separate study is required.

In this work we propose two algorithms: (i) the novel graph traversing algorithm that allows to avoid backtracking and reveal the structure of the graph between any two nodes and (ii) the semi-analytical algorithm based on representation of non-Gaussians distributions by the linear combination of kernel functions that goes beyond existing results and allows further generalisation. The key outcome is: (i) the *fully analytical expression* for the convolution of $\max$ of two Gaussians with the gate and/or interconnect delay, and (ii) the methodology of handling *non-Gaussian distributions* so that obtained result (3)–(5) becomes applicable.

The new graph traversal algorithm combined with kernel function decomposition allows one a straightforward extension of the analysis to the key challenges of current SSTA: spatial correlations, multiple inputs to one gate and all-non-Gaussian distributions of gate delays. We are currently working to incorporate these modifications and this is our on-going research.

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. The authors would like to thank also Steve Hollands, Paul Frain, Adrian Wrixon and Kelvin Le for stimulating discussions.

References

[1] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs. A Practical Approach*. Springer, 2009.

[2] L. Lavagno, I. L. Markov, G. Martin, and L. K. Scheffer, Eds., *Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology*. CRC Press, 2016.

[3] C. Y. Lee and N. K. Jha, "Fincanon: A pvt-aware integrated delay and power modeling framework for finfet-based caches and on-chip networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 5, pp. 1150–1163, May 2014.

[4] A. Tang and N. K. Jha, "Genfin: Genetic algorithm-based multiobjective statistical logic circuit optimization using incremental statistical analysis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 3, pp. 1126–1139, Mar 2016.

[5] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical timing anlaysis: From basic principles to state of the art," *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, vol. 4, no. 8, 2008.

[6] J. Chung and J. A. Abraham, "Concurrent path selection algorithm in statistical timing analysis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 9, pp. 1715–1726, Sept 2013.

[7] A. Lange, C. Sohrmann, R. Jancke, J. Haase, B. Cheng, A. Asenov, and U. Schlichtmann, "Multivariate modeling of variability supporting non-gaussian and correlated parameters," *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, vol. 35, no. 2, pp. 197–210, Feb 2016.

[8] V. Rao, D. Sinha, N. Srimal, and P. K. Maurya, "Statistical path tracing in timing graphs," in *Proc. DAC*, Jun 2016, pp. 1–6.

[9] I. Kovacs, M. opa, A. Buzo, and G. Pelz, "An accurate yield estimation approach for multivariate non-normal data in semiconductor quality analysis," in *Proc. SMACD*, Jun 2017, pp. 1–4.

[10] S. H. Gerez, Ed., *Algorithms for VLSI Design Automation*. Wiley, 1998.

[11] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, S. Narayan, D. K. Beece, J. Piaget, N. Venkateswaran, and J. G. Hemmett, "First-order incremental block-based statistical timing analysis," *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2170–2180, Oct. 2006.

[12] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*, 4th ed. McGraw-Hill, 2002.

[13] S. Nadarajah and S. Kotz, "Exact distribution of the max/min of two gaussian random variables," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 2, pp. 210–212, Feb 2008.

[14] C. E. Clark, "The greatest of a finite set of random variables," *Oper. Res.*, vol. 9, no. 2, pp. 145–162, Apr. 1961.