Variation-Aware Global Placement for Improving Timing-Yield of Carbon-Nanotube Field Effect Transistor Circuit

CHEN WANG and YANAN SUN, Shanghai Jiao Tong University
SHIYAN HU, Michigan Technological University
LI JIANG and WEIKANG QIAN, Shanghai Jiao Tong University

As the conventional silicon-based CMOS technology marches toward the sub-10nm region, the problem of high power density becomes increasingly serious. Under this circumstance, the carbon-nanotube field effect transistors (CNFETs) emerge as a promising alternative to the conventional silicon-based CMOS devices. However, they experience a much larger variation than the silicon-based CMOS devices, which results in a large circuit delay variation and hence, a significant timing yield loss. One of the main variation sources is the carbon-nanotube (CNT) density variation. However, it shows a special property not existing for silicon-based CMOS devices, namely the asymmetric spatial correlation. In this work, we propose novel global placement algorithms to reduce the timing yield loss caused by the CNT density variation. To effectively reduce the statistical circuit delay, we first develop a statistical delay measure for a segment of gates. Based on this measure, we further develop a segment-based strategy and a path-based placement strategy to reduce the delays of the statistically critical paths. Experimental results demonstrated that both of our approaches effectively improve the timing yield.

CCS Concepts: • **Hardware → Placement**; **Emerging tools and methodologies**; *Software tools for EDA*;

Additional Key Words and Phrases: Carbon-nanotube field effect transistor (CNFET) circuit, variation-aware placement, statistical static timing analysis (SSTA), statistical timing optimization, timing-yield improvement

ACM Reference format:
Chen Wang, Yanan Sun, Shiyan Hu, Li Jiang, and Weikang Qian. 2018. Variation-Aware Global Placement for Improving Timing-Yield of Carbon-Nanotube Field Effect Transistor Circuit. *ACM Trans. Des. Autom. Electron. Syst.* 23, 4, Article 44 (June 2018), 27 pages.
https://doi.org/10.1145/3175500

This work is supported in part by National Natural Science Foundation of China (NSFC) under Grant Nos. 61602300, 61704104, and 61472243, Shanghai Science and Technology Committee under Grant No. 15YF1406000, Shanghai Sailing Program under Grant No. 17Z111260004, SJTU Medical and Engineering Interdisciplinary Funding under Grant No. YG2015MS17, and U.S. National Science Foundation (NSF) CAREER Award CCF-1349984.

Authors' addresses: C. Wang and W. Qian, University of Michigan-Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University, Shanghai, China; Y. Sun, Department of Micro/Nano Electronics, Shanghai Jiao Tong University, Shanghai, China; S. Hu, Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI, USA; L. Jiang, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China.
Authors' emails: C. Wang, wangchen_2011@sjtu.edu.cn; Y. Sun, sunyanan@sjtu.edu.cn; S. Hu, shiyan@mtu.edu; L. Jiang, ljiang_cs@sjtu.edu.cn; W. Qian, qianwk@sjtu.edu.cn.
The corresponding authors: L. Jiang and W. Qian.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 ACM 1084-4309/2018/06-ART44 $15.00
https://doi.org/10.1145/3175500

1 INTRODUCTION

As the conventional silicon-based CMOS technology marches toward the sub-10nm region, we are approaching the ultimate physical limit of the silicon-based CMOS device. Under this circumstance, many emerging devices are being aggressively investigated. One such device is the carbon-nanotube field effect transistors (CNFETs). The device structure of a CNFET is shown in Figure 1(a). It is similar to the conventional silicon-based MOSFET, except that the channel of a CNFET is composed of a bundle of carbon-nanotubes (CNTs). The major advantages of the CNFET devices over the silicon-based MOSFET devices are the larger driving current density and smaller power consumption [7]. It has been reported that a CNFET circuit can have a potential improvement of the energy-delay product, which is regarded as a metric of the energy efficiency, by more than an order of magnitude over a conventional silicon-based CMOS circuit [19].

However, there are several challenges to be overcome to design very large-scale CNFET circuits [10, 23, 37]. The main challenges include the density variation of CNTs, the existence of metallic CNTs, and the misalignment of CNTs. The recent advances in the CNT synthesis process and the design methodologies for CNFET circuits were able to solve the problems of the existence of metallic CNTs [15, 21] and the misalignment of CNTs [18, 22]. However, the CNT density variation still remains as a major problem that affects both the reliability and the performance of the CNFET circuits, as modeled and analyzed in References [26, 34].

The CNT density variation is due to the lack of precise control during the manufacturing of CNTs [12, 39]. An example of the CNT alignment is shown in Figure 1(b), from which we can see that the distance between any two adjacent CNTs varies significantly. A direct consequence of the density variation is that the driving current of a CNFET has a large variation, since the current is proportional to the number of CNTs covered under the gate of the CNFET. As a result, CNFET circuits have a large delay variation [26, 38], leading to a degraded timing yield [9, 12].

A few previous works explored placement techniques to overcome the problem of the large circuit delay variation caused by CNT density variation. These works exploited a special property of the CNFET circuit. As we can see from Figure 2, if two CNFETs are aligned in parallel to the CNT growth direction, which means that their channels cover the same bundle of CNTs, then their electrical properties are highly correlated. On the other hand, if the channels of two CNFETs cover different bundles of CNTs, then their electrical properties are highly independent. This special property is known as **asymmetric spatial correlation** [39]. In Reference [1], the authors proposed a path-healing method that distributes the gates on each near critical path to different columns along the CNT growth direction during the detailed placement phase. By this approach, the variation of a path delay is effectively reduced. However, since this method only moves the gates during the detailed placement phase, the optimization space is limited. Furthermore, the authors did not consider the interconnect delay, i.e., the delay caused by the parasitics of the interconnect wires, which, according to our study, contributes a large portion to the circuit delay variation. In our previous work [31], we proposed a timing-driven global placement technique to optimize the location of the CNFET gates along the critical paths. We introduced a new distribution force to the force-directed placement framework. The interconnect delay is considered. However, while the distribution force moves the gates to reduce the variance of the path delay, it may also increase the mean value of the path delay and, thus, degrade the timing yield.

In this work, to address the drawbacks of the previous works, we propose novel variation-aware global placement approaches for CNFET circuits. The proposed methods take into consideration both the mean and the variance of the gate delay and the interconnect delay. To effectively optimize the statistical circuit delay, we first develop a first-order approximation of the standard deviation of the stage delay, where a stage is composed of a single gate and the net driven by the gate. Based on the stage delay approximation, we further develop a statistical delay measure for a segment

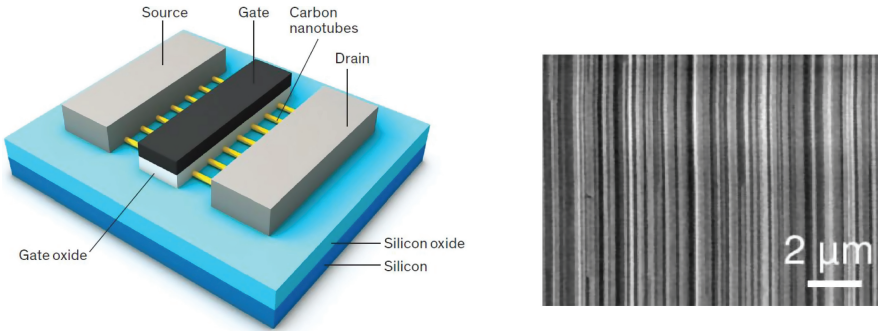

Fig. 1. CNFET structure and CNTs: (a) a 3D view of CNFET [27]; (b) aligned CNT growth [20].

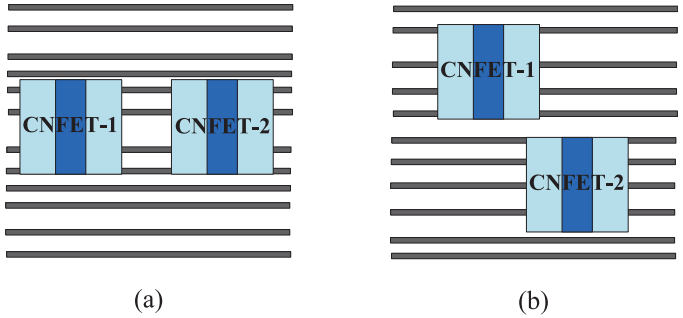

Fig. 2. An illustration of the asymmetric spatial correlation of CNFET circuits: (a) two CNFETs are highly correlated together if they are aligned along the CNT growth direction; (b) two CNFETs are almost uncorrelated if they are not aligned along the CNT growth direction.

of gates. This delay measure is composed of both the mean and the standard deviation of the delay for that segment. Based on the statistical delay measure, we propose the first variation-aware global placement method, which is a segment-based method. It identifies the statistically critical segments of gates in a circuit and then optimizes the statistical delay measure for each segment. With some modifications to the segment-based method, we propose the second global placement method, which is path-based. Experimental results showed that the proposed methods significantly improve the circuit timing yield.

In summary, the main contributions of our work are as follows.

- We propose a novel statistical delay measure for a segment of gates in the CNFET circuit. The measure is composed of both the mean and the standard deviation of the segment delay, where the standard deviation is derived from a first-order approximation of the standard deviation of the stage delay. It takes the asymmetric spatial correlation of the CNT density variation into consideration.
- We propose a segment-based variation-aware global placement algorithm to improve the timing yield of the CNFET circuit. It applies a segment-based optimization technique to relocate the gates in a set of statistically critical segments. It optimizes the statistical delay measures for those critical segments.
- With some changes to the segment-based approach, we also propose a path-based variation-aware global placement method.

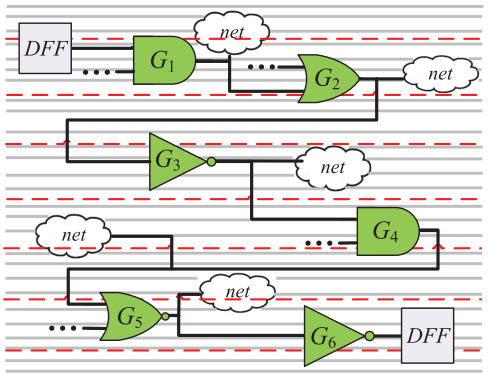

Fig. 3. An example of a global placement of six CNFET gates along a path. The horizontal grey lines are parallel CNTs. The red dashed lines indicate the boundaries of the rows in the row-based standard cell placement.

The remainder of this article is organized as follows. In Section 2, we introduce the background on CNT density variation, CNFET standard cell library, and the functional yield and the timing yield of CNFET circuit. In Section 3, we present our timing model for a CNFET standard cell. Then, based on the timing model for a standard cell, in Section 4, we present our timing model for a stage in the circuit, which is composed of a gate and the net driven by the gate. Then, based on our timing model for a stage, in Section 5, we present our statistical delay measure for a segment. In Sections 6 and 7, we elaborate our proposed segment-based and path-based variation-aware global placement algorithms, respectively. In Section 8, we present the experimental results. Finally, we conclude the article in Section 9.

2 BACKGROUND

In this section, we introduce the background on CNT density variation, CNFET standard cell library used in our study, and the functional yield and the timing yield of the CNFET circuits. Along this introduction, we show the important process and the design parameters we choose in this study. We also show that with the chosen parameters, the functional yield is not a major concern and, thus, it is meaningful to improve the timing yield of the CNFET circuits.

2.1 CNT Density Variation

Because the CNTs are self-assembled without the exact control of their locations [39], CNT density, defined as the number of CNTs per unit width, varies at different locations on a chip. However, different from the silicon-based CMOS technology, this variation shows a strong **asymmetric spatial correlation**. For different locations along the CNT growth direction, the CNT densities are highly correlated, while for the locations not along the CNT growth direction, the CNT densities are highly independent. Figure 3 illustrates this special pattern of CNT density variation. In the figure, the CNTs are represented by grey lines. As can be seen from the figure, the CNT densities do not change along the horizontal direction, while they vary along the vertical direction.

The CNT density variation depends on the CNT process parameters. In this work, we use the CNFET process parameters from the recent literatures. The major CNT process parameters are listed in Table 1.

The ideal CNTs used in the channel should be semiconducting. However, due to the imperfection of the manufacturing process, some CNTs are metallic. We assume that the probability of a CNT

Table 1. CNT Process Parameters Appearing in Recent Literatures [11, 19] and the Values Used in this Work

parameter	meaning	value in references	our choice
μ_s	mean value of inter-CNT pitch before the removal of metallic CNTs	4nm [19]	4nm
IDC	index of dispersion for CNT count	0.5 [38]	0.5
p_m	probability that a CNT is metallic	≤ 0.01 [17]	0.01
p_{Rs}	conditional probability that a CNT is removed given that it is semiconducting	≤ 0.05 [21]	0.05
p_{Rm}	conditional probability that a CNT is removed given that it is metallic	≥ 0.9999 [21]	1
ϕ_{CNT}	diameter of each CNT	[1.2nm, 1.8nm] [19]	1.3nm

to be metallic is $p_m = 0.01$ [17]. Fortunately, there exist several effective methods to remove up to 99.99% metallic CNTs [21, 35]. Therefore, in this work, we assume that *all* the metallic CNTs are removed and $p_{R_m} = 1$.

The work in Reference [38] shows that the CNT count covered by a given width W, $n(W)$, approximately follows a normal distribution of the form $N(\frac{W}{\mu_s}, \frac{W\sigma_s^2}{\mu_s^3})$, where μ_s and σ_s are the mean and standard deviation of the distance between two adjacent CNTs.

By the metallic CNT removal procedure, all the metallic CNTs are removed and a small portion of the semiconducting CNTs are also removed. What we are ultimately interested in is the distribution of the *semiconducting* CNT count after the metallic CNT removal process. The semiconducting CNT count covered by a given width W, $n_s(W)$, also approximately follows a normal distribution of the form

$$n_s(W) \sim N\left(\frac{W}{\mu_{s,post}}, \frac{W\sigma_{s,post}^2}{\mu_{s,post}^3}\right), \tag{1}$$

where $\mu_{s,post}$ and $\sigma_{s,post}$ are the mean and standard deviation of the distance between two adjacent semiconducting CNTs after the metallic CNT removal process.

By using the parameters from Table 1 and the method proposed in Reference [39], we can derive that

$$\mu_{s,post} = 4.253\text{nm}, \sigma_{s,post} = 3.085\text{nm}. \tag{2}$$

Since in our study, we assume the non-existence of the metallic CNTs due to the metallic CNT removal process, all the CNTs are semiconducting. In what follows, when we say a CNT, it refers to a semiconducting CNT.

2.2 CNFET Standard Cell Library

A standard cell library provides the basic building blocks for designing a digital circuit. Therefore, to design CNFET circuits, a CNFET standard cell library is needed. Since there is no mature CNFET standard cell library available, we use the *Nangate* 45nm standard cell library [16] as the reference to construct the CNFET standard cells.

In our study, we consider two technology nodes, 14nm and 10nm. Some important design parameters for our 14nm and 10nm CNFET standard cell libraries are listed in Table 2. For each standard cell in our 14nm/10nm CNFET standard cell library, we design its schematic, its cell dimensions, and its HSPICE model. The schematic of each cell is designed with reference to the *Nangate* 45nm standard cell library [16]. The dimension of each cell is scaled from the reference cell in the *Nangate* 45nm standard cell library by a ratio of the standard cell height in Table 2 (i.e., $0.768\mu m$ and $0.679\mu m$ for 14nm and 10nm technology nodes, respectively) over the standard cell height of the *Nangate* library (i.e., $1.4\mu m$). We obtain the HSPICE model for each standard cell by

Table 2. Parameters for Both 14nm and 10nm Technology Nodes Used in Our Study

parameter	value	
technology node (nm)	14	10
V_{dd} (V)	0.5	0.5
CNFET gate length L_g (nm)	9.8	9.5
CNFET contact length L_c (nm)	44.3	38.1
CNFET external length L_{ext} (nm)	4.95	4.5
minimum CNFET gate width (nm)	32	28.8
maximum CNFET gate width (nm)	280	240
CNFET flat band voltage (V)	0.007	0.01
CNFET gate oxide thickness T_{ox} (nm)	2.57	2.51
CNFET gate oxide dielectric constant K_{ox}	13	14
dielectric constant of spacer K_{spa}	2.775	2.59
standard cell height (μm)	0.768	0.679
standard cell width (relative)	1	0.884
CNT diameter (nm)	1.3	1.3
$\mu_{s,post}$ (nm)	4.253	4.253
$\sigma_{s,post}$ (nm)	3.085	3.085
interconnect unit capacitance (fF/μm)	0.175	0.165
interconnect unit resistance (Ω/μm)	23.746	29.363

The resistance and the capacitance for unit length of *Cu* interconnect are derived from ITRS2013 report [14]. $\mu_{s,post}$ and $\sigma_{s,post}$ are the mean and the standard deviation of the inter-CNT pitch of the remaining semiconducting CNTs after applying the metallic CNT removal process. Their values are from Equation (2). Other parameters are taken from Reference [12].

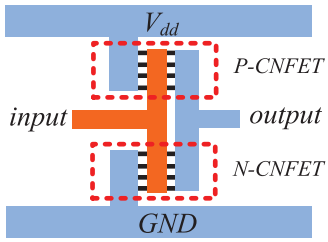

Fig. 4. A CNFET inverter. N-type CNFET and P-type CNFET use different bundles of CNTs as their channels.

applying the Stanford University Virtual Source CNFET HSPICE model [30] to each CNFET that constitutes the entire standard cell.

According to the layout style of the *Nangate* library, the gate channel is in parallel to the V_{dd}/GND rail. Consequently, for each CNFET cell in our library, the growth direction of CNTs is in parallel to the row direction along which the standard cells are placed. A CNFET inverter of this layout style is shown in Figure 4. Note that the P-type and N-type CNFETs cover different bundles of CNTs.

The recent work in Reference [2] shows that by using an aligned-active layout style, the functional yield of a CNFET circuit can be improved. Therefore, we also apply the aligned-active layout style in designing our CNFET standard cells. In the aligned-active layout style, the CNFETs in the pull-up network (PUN) are correlated together as much as possible and the same for the CNFETs in the pull-down network (PDN). The widths of the CNFETs in each standard cell are determined

to let the PUN and PDN of the cell have the same delay when the number of CNTs of each CNFET is chosen as the nominal value.

According to Table 2, the minimum width for a CNFET in either the PUN or the PDN is set as 32nm and 28.8nm for the 14nm and 10nm technology nodes, respectively. These minimum widths are able to provide a relatively high functional yield for the CNFET circuits, which we will show in Section 2.3. The maximum CNFET width is set as 280nm and 240nm for the 14nm and 10nm technology nodes, respectively. Given the maximum CNFET width, when we build a CNFET standard cell with a large driving strength, if the effective width of a CNFET exceeds the maximum value, then we apply transistor folding to build that CNFET.

2.3 Functional Yield and Timing Yield

The CNT density variation introduced in Section 2.1 affects both the functional yield and timing yield of the CNFET circuits. **Functional yield** is the rate of the manufactured chips that function correctly. **Timing yield** is the rate of the manufactured chips that satisfy a given circuit delay constraint.

Due to the CNT density variation, it is possible that a CNFET contains zero CNT. Since a CNFET uses CNTs as the channel, a CNFET with zero CNT will fail to function correctly. If a circuit has at least one CNFET with zero CNT, then the circuit has a functional failure. Therefore, the CNT density variation could cause the functional yield loss for the CNFET circuits. On the other hand, for a functional correct circuit, due to the CNT density variation, its delay could be larger than the required value. Therefore, the CNT density variation could also cause the timing yield loss for the CNFET circuits.

To design practical CNFET circuits, the first target is to guarantee a sufficiently high functional yield. This can be realized by choosing a sufficiently large value for the minimum CNFET width. As we mentioned in Section 2.2, the minimum CNFET width is set as 32nm and 28.8nm for 14nm and 10nm technology nodes, respectively. In our study, the largest benchmark is the "M1-CORE" from Reference [24]. By the functional yield analysis method in Reference [36], under the chosen minimum CNFET width and the CNT process parameters in Table 1, the functional yields of this benchmark are at least 95.2% and 90.0% for the 14nm and 10nm technology nodes, respectively. For the other smaller benchmarks used in our study, their functional yields are even higher. Therefore, choosing the minimum CNFET width as 32nm and 28.8nm for the 14nm and 10nm technology nodes, respectively, ensures a relatively high functional yield and, hence, the functional yield is not a major problem. Under this circumstance, our major target is to optimize the timing yield of a CNFET circuit.

3 GATE DELAY MODELS FOR CNFET STANDARD CELLS

To evaluate the timing of a CNFET circuit, the timing information for each CNFET standard cell must be characterized. In this section, we first develop the deterministic delay models for the CNFET standard cells, in which we assume that the CNT counts in the PUN and PDN of CNFET standard cells are fixed. Then, based on the deterministic models, we develop statistical delay models for the CNFET standard cells, in which we assume that the CNT counts are random. In both the deterministic and the statistical delay models, we characterize the gate intrinsic delay d_0, the output resistance R_{out}, the input slew rate coefficient k, and the input capacitance C_{in} for each CNFET standard cell in the library.

3.1 Deterministic Gate Delay Model

In this section, we develop the deterministic gate delay models for the CNFET standard cells. They will be used later to obtain the statistical gate delay models for the CNFET cells.

For a conventional CMOS gate, the work in Reference [6] presents the following gate delay model:

$$d_{gate} = d_0 + R_{out} \cdot C_{load} + k \cdot S_{in}, \tag{3}$$

where d_0 is the intrinsic gate delay, R_{out} is the output resistance of the gate, C_{load} is the load capacitance at the output of the gate, k is a constant coefficient, and S_{in} is the input slew rate. Note that the effect of the self-loading capacitance of the standard cell on its delay is included in the intrinsic gate delay d_0. C_{load} is the total loaded capacitance at the output of the standard cell excluding its self-loading capacitance. Thus, C_{load} is independent of its driving standard cell.

In our work, we also apply Equation (3) to model the gate delay of a CNFET standard cell. The values d_0, R_{out}, and k are intrinsic to each specific CNFET standard cell. Furthermore, since the number of CNTs in the channel of a CNFET affects the driving current and the delay of the CNFET, the values of d_0, R_{out}, and k also depend on the CNT count. To obtain the deterministic gate delay model for each CNFET standard cell, we characterize the values of d_0, R_{out}, and k for different CNT counts, and the characterization results are stored in look-up tables.

Indeed, depending on whether the PUN or the PDN of the CNFET cell is turned on, the values d_0 and R_{out} are related to either the CNT count in the PUN or that in the PDN. Therefore, we further distinguish the intrinsic gate delay into the one when the PUN is on, $d_{0,PUN}$, and the one when the PDN is on, $d_{0,PDN}$. Similarly, we distinguish the output resistance into the one when the PUN is on, $R_{out,PUN}$, and the one when the PDN is on, $R_{out,PDN}$. For each CNFET standard cell, we characterize $d_{0,PUN}$ and $R_{out,PUN}$ for different CNT counts in the PUN of the cell; we characterize $d_{0,PDN}$ and $R_{out,PDN}$ for different CNT counts in the PDN of the cell. Based on our characterization results, we find that the value of k is almost independent of the CNT counts in the PUN and PDN of the cell. Therefore, we characterize the coefficient k by assuming the nominal CNT counts in the PUN and PDN of the cell. Here, the nominal CNT count is obtained by assuming that the distance between any two adjacent CNTs is $\mu_{s,post} = 4.253$nm (see Equation (2)).

We simulate the HSPICE model of each CNFET standard cell, which is described in Section 2.2, and obtain the constant coefficient k and the relations of $d_{0,PUN}$, $d_{0,PDN}$, $R_{out,PUN}$, and $R_{out,PDN}$ versus CNT count for each CNFET standard cell. The results are stored in the corresponding look-up tables.

To use the deterministic gate delay model, when we are given a specific CNFET standard cell, the CNT counts in its PUN and PDN, the input slew rate S_{in}, and the output capacitance C_{load}, we can obtain the gate delay as

$$d_{gate} = \max\left\{d_{0,PUN} + R_{out,PUN} \cdot C_{load} + k \cdot S_{in}, \quad d_{0,PDN} + R_{out,PDN} \cdot C_{load} + k \cdot S_{in}\right\}, \tag{4}$$

where the values $d_{0,PUN}$, $d_{0,PDN}$, $R_{out,PUN}$, $R_{out,PDN}$, and k are obtained from the look-up table based on the cell type and the CNT counts. Note that in our work, we perform the topological timing analysis, ignoring the logic functions of the gates. Therefore, the gate delay is obtained as the larger one between the delay of the PUN and the delay of the PDN.

Another important parameter that we characterize for each standard cell is the input capacitance C_{in}. This contributes to the load capacitance C_{load} of each driving gate and hence, the timing of a CNFET circuit. To characterize the input capacitance of a cell, for an input pin of the cell, we obtain the CNFETs controlled by that pin. Then, the total input capacitances of these CNFETs in the PUN of the standard cell for different CNT counts in the PUN are obtained numerically by using the Stanford University Virtual Source CNFET Matlab Model [30]. The same is done to obtain the total input capacitances of the CNFETs for different CNT counts in the PDN. To use the input capacitance characterization result, when we are given a specific CNFET standard cell and the CNT counts in its PUN and PDN, we can obtain the input capacitance of the standard cell as the sum of the total input capacitances in its PUN and PDN for the given CNT counts.

3.2 Statistical Gate Delay Model

To implement the statistical static timing analysis (SSTA) to guide the statistical timing optimization in this study, we establish the statistical timing model for the cells in our CNFET standard cell library.

In the delay model shown in Equation (4), C_{load} is a variable independent of the specific CNFET standard cell. The existence of the variable C_{load} in the max operator makes the further mathematical handling difficult. For simplification, we approximate the gate delay shown in Equation (4) by the following equation:

$$d_{gate} = \max\{d_{0,PUN}, d_{0,PDN}\} + \max\{R_{out,PUN}, R_{out,PDN}\} \cdot C_{load} + k{\cdot}S_{in}. \tag{5}$$

It can be seen that the value calculated by Equation (5) is always an upper bound of the value calculated by Equation (4). We further define

$$d_{0,max} = \max\{d_{0,PUN}, d_{0,PDN}\}, R_{out,max} = \max\{R_{out,PUN}, R_{out,PDN}\}. \tag{6}$$

Then, the gate delay shown in Equation (5) can be further represented as

$$d_{gate} = d_{0,max} + R_{out,max} \cdot C_{load} + k{\cdot}S_{in}. \tag{7}$$

For simplicity, in what follows, we refer to $d_{0,max}$ as the **intrinsic gate delay** and $R_{out,max}$ as the **output resistance** of a given gate.

In the statistical gate delay model, $d_{0,max}$ and $R_{out,max}$ are random variables. As we will show later, our proposed statistical timing model for a CNFET circuit depends on the mean and the standard deviation of the random variables $d_{0,max}$ and $R_{out,max}$ for each CNFET standard cell. To characterize these values, for each CNFET standard cell, we randomly generate 10,000 sets of the CNT counts in the PUN and the PDN based on the normal distribution shown in Equation (1). For each set of the CNT counts, we first obtain the values $d_{0,PUN}$, $d_{0,PDN}$, $R_{out,PUN}$, and $R_{out,PDN}$ through the look-up table and then obtain the values $d_{0,max}$ and $R_{out,max}$ by Equation (6). Then, the mean and the standard deviation of $d_{0,max}$ ($R_{out,max}$) are obtained as the mean and the standard deviation of the 10,000 samples of $d_{0,max}$ ($R_{out,max}$).

In the statistical timing model, the input capacitance C_{in} is also a random variable. As we will show later, our proposed statistical timing model for a CNFET circuit also depends on the mean and the standard deviation of the random variable C_{in} for each CNFET standard cell. We also use random simulation together with table look-up to obtain the mean and the standard deviation of C_{in} for each CNFET standard cell.

4 STAGE DELAY MODEL

Based on the delay models presented in the Section 3, in this section, we present the stage delay model, which will be used in Section 5 below to obtain the statistical delay measure for a segment. The stage delay is defined as the delay from the input of a driving gate to the input of a driven gate. It is evaluated as the sum of the delay of the driving gate and the delay caused by the interconnect wire. It gives the delay of one stage of a path in the circuit. In this section, we first describe how we model the stage delay. Then, we show simplified approximations to the mean and the standard deviation of a stage delay, which will play an important role for calculating the statistical delay measure for a segment.

4.1 Modeling Stage Delay

We use the example shown in Figure 5(a) to illustrate the modeling of the stage delay. Suppose that the driving gate is the gate G_i and it drives m_i gates $G_{i_1}, \ldots, G_{i_{m_i}}$, where $i_1, \ldots, i_{m_i}$ are the indices of these m_i gates. We evaluate the stage delay from the input of the driving gate G_i to the input

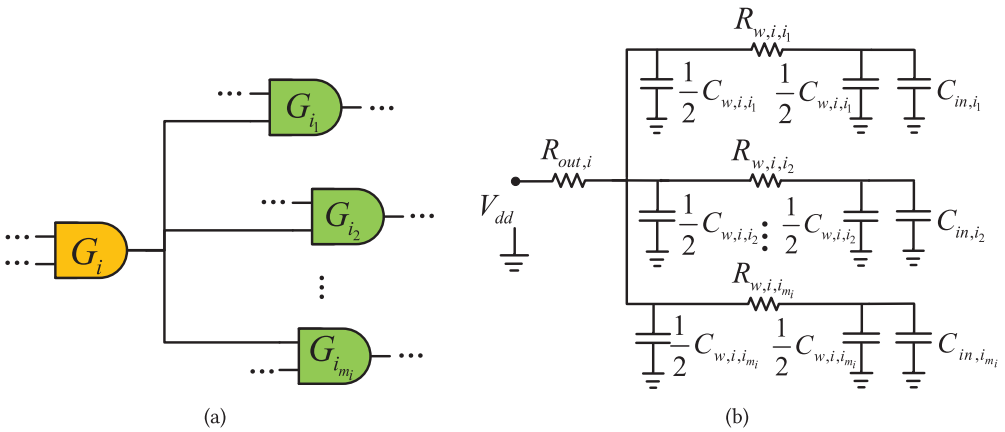

Fig. 5. An example to illustrate the stage delay modeling: (a) a CNFET gate driving a number of gates; (b) the RC tree for the interconnect in (a).

of the gate G_{i_l} for a specific $1 \le l \le m_i$. The interconnect wires between the driving gate and the driven gates are modeled as an RC tree shown in Figure 5(b), in which each wire connecting the driving gate and a driven gate is represented using a π-model [32]. The delay caused by the output resistance of the driving gate and the interconnect wire is obtained by applying the Elmore delay model to the RC tree [8]. Then, the stage delay is the sum of the gate intrinsic delay, the delay caused by the input slew, and the delay caused by the output resistance of the driving gate and the interconnect wire. For this example, the stage delay is

$$d_{st,i} = d_{0,max,i} + k_i \cdot S_{in,i} + R_{out,max,i} \cdot \left(\sum_{j=1}^{m_i} C_{w,i,i_j} + C_{in,i_j} \right) + R_{w,i,i_l} \cdot \left(\frac{1}{2} C_{w,i,i_l} + C_{in,i_l} \right), \quad (8)$$

where $d_{0,max,i}$, k_i, $S_{in,i}$, and $R_{out,max,i}$ are the intrinsic delay, input slew rate coefficient, input slew rate, and the output resistance, respectively, of the driving gate G_i, C_{w,i,i_j} and R_{w,i,i_j} are the parasitic capacitance and resistance, respectively, of the interconnect wire between the driving gate G_i and the driven gate G_{i_j}, and C_{in,i_j} is the input capacitance of the driven gate G_{i_j}.

The parasitic capacitance $C_{w,i,j}$ and resistance $R_{w,i,j}$ of the interconnect wires are calculated as

$$C_{w,i,j} = c_{unit} \cdot HPWL_{i,j}, \; R_{w,i,j} = r_{unit} \cdot HPWL_{i,j},$$

where c_{unit} and r_{unit} are the parasitic capacitance and resistance, respectively, per unit length of the interconnect wire, and $HPWL_{i,j}$ is the half perimeter wire-length (HPWL) between the gates G_i and G_j [6]. Since during the global placement, the routing is not done yet, the precise information of the interconnect wire is not available. Therefore, we can only estimate the interconnect parasitic parameters. Since HPWL is widely used for delay estimation during the global placement [6, 29], we use it to estimate the interconnect wire length in our stage delay timing model.

4.2 Mean and Standard Deviation of Stage Delay

In this section, we show the simplified approximations to the mean and the standard deviation of the stage delay used in our work. They are used later on for simplifying both the evaluation of the proposed statistical delay measure for a segment and the statistical static timing analysis. Nevertheless, they are accurate enough for providing a guideline in our optimization flow.

We approximate the mean of the stage delay by the nominal value of the stage delay, i.e.,

$$\mu\left(d_{st,i}\right) \approx d_{st,i}^{nom}, \tag{9}$$

where $d_{st,i}^{nom}$ is calculated by Equation (8) using the mean values of the parameters $d_{0,max,i}$, $R_{out,max,i}$, and $C_{in,i}$ characterized by the method described in Section 3.2.

Next, we show how we obtain an approximation to the standard deviation of the stage delay. According to the expression for the stage delay in Equation (8), we can obtain the standard deviation of the stage delay as

$$\begin{aligned}\sigma_1 &= \sigma(d_{st,i}) \\ &= \sigma\left(d_{0,max,i} + k_i \cdot S_{in,i} + R_{out,max,i} \cdot \left(\sum_{j=1}^{m_i} C_{w,i,i_j} + C_{in,i_j}\right) + R_{w,i,i_l} \cdot \left(\frac{1}{2} C_{w,i,i_l} + C_{in,i_l}\right)\right).\end{aligned} \tag{10}$$

In the above equation, the intrinsic gate delay $d_{0,max,i}$ and output resistance $R_{out,max,i}$ are random variables depending on the CNT counts in the driving gate. The input capacitances C_{in,i_j}'s are random variables depending on the CNT counts in the driven gates. The other variables in the equation are constants.

In our statistical timing characterization of the CNFET standard cells, we found that the standard deviation of the output resistance of the driving gate (i.e., $\sigma(R_{out,max,i})$) is much larger than those of the intrinsic gate delay of the driving gate and the input capacitances of the driven gates (i.e., $\sigma(d_{0,max,i})$ and $\sigma(C_{in,i_j})$). Therefore, we propose the following approximation to the standard deviation of the stage delay,

$$\sigma_2 = \sigma\left(R_{out,max,i}\right) \cdot \left(\sum_{j=1}^{m_i} C_{w,i,i_j} + \mu\left(C_{in,i_j}\right)\right), \tag{11}$$

where $\mu(C_{in,i_j})$ is the mean value of the input capacitance C_{in,i_j} characterized by the method described in Section 3.2.

To validate that the σ_2 shown in Equation (11) is a close approximation to the σ_1 shown in Equation (10), we did Monte Carlo simulation on the benchmark "PID controller" from Reference [24], which has 3,102 nets. We used 14nm CNFET standard cell library. We performed a global placement to obtain the location of each gate in the benchmark. For each net in the benchmark, we randomly generated 1,000 sets of CNT counts for the driving and driven gates and calculated the standard deviations σ_1 and σ_2 based on these 1,000 samples. Then we obtained the mean values $\mu(\cdot)$ and standard deviations $\sigma(\cdot)$ of the σ_1's and σ_2's over all 3,102 nets. The results are

$$\mu(\sigma_1) = 7.0017, \sigma(\sigma_1) = 26.9360, \tag{12}$$

$$\mu(\sigma_2) = 6.7920, \sigma(\sigma_2) = 26.9345. \tag{13}$$

It can be seen that the mean and the standard deviation of σ_2 are very close to the respective values of σ_1. Furthermore, we also computed the absolute error defined as $|\sigma_2 - \sigma_1|$ for each net. The mean and standard deviation of the absolute errors over all the nets are 0.2100 and 0.1720, respectively. These values are very small compared to the mean and standard deviation of σ_1 shown in Equation (12). This indicates that σ_2 is a close approximation to σ_1.

Therefore, we conclude that

$$\sigma\left(d_{st,i}\right) \approx \sigma\left(R_{out,max,i}\right) \cdot \left(\sum_{j=1}^{m_i} C_{w,i,i_j} + \mu\left(C_{in,i_j}\right)\right) = \sigma\left(R_{out,max,i}\right) \cdot C_{load,i}^{nom}, \tag{14}$$

where we define

$$C_{load,i}^{nom} = \sum_{j=1}^{m_i} C_{w,i,i_j} + \mu\left(C_{in,i_j}\right).$$

Notice that the output resistance $R_{out,max,i}$ of the driving gate is obtained by our statistical gate delay characterization method for each individual CNFET standard cell, which is described in Section 3.2. Equation (14) also indicates that the variation of the output resistance of the driving gate dominates the variation of the stage delay.

In summary, Equations (9) and (14) give approximations to the mean and the standard deviation of the stage delay, respectively. They will be used later in our statistical measure of segment delay.

5 STATISTICAL DELAY MEASURE FOR A SEGMENT

A segment is a sequence of connected gates as part of a path in the circuit. In this section, we introduce a statistical delay measure for a segment, which will be used in our statistical timing optimization process. The calculation of the statistical delay measure depends on the correlation matrix for a CNFET circuit, which we introduce next.

5.1 Correlation Matrix for CNFET Circuits

CNFET circuits have a larger spatial variation than conventional silicon-based CMOS circuits. However, the variation of a CNFET circuit exhibits the unique asymmetric spatial correlation, as we mentioned in Section 2.1. This feature provides us with an opportunity to explore new techniques to reduce the performance degradation of a CNFET circuit caused by the CNT density variation.

In our work, we characterize the spatial variation of a CNFET circuit using a correlation matrix M. Given a set of N CNFET gates $G_1, \ldots, G_N$ and a global placement of them, the correlation matrix is an N-by-N matrix constructed as follows. For two CNFET gates G_i and G_j, if their geometric centers belong to the same row of the row-based standard cell placement, we treat them as completely correlated and set the entries $M_{i,j}$ and $M_{j,i}$ as 1, where $M_{i,j}$ denotes the entry at row i and column j of the matrix M. Otherwise, the entries $M_{i,j}$ and $M_{j,i}$ are set as 0. The entries $M_{i,i}$'s are all set as 1. We illustrate how to build the correlation matrix in the following example.

Example 5.1. Consider the global placement shown in Figure 3. The CNTs, indicated by the grey lines in the figure, are grown horizontally. The red dashed lines indicate the boundaries of the rows in the row-based standard cell placement. The gates G_1 and G_2 have their geometric centers in the same row and the same for the gates G_5 and G_6. Therefore, the entries $M_{1,2} = M_{2,1} = M_{5,6} = M_{6,5} = 1$. However, the geometric centers of the gates G_3 and G_4 are at different rows. Therefore, the entries $M_{3,4} = M_{4,3} = 0$. The entire correlation matrix for these 6 gates is

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

5.2 Statistical Delay Measure for a Segment

In this section, we present our proposed statistical delay measure for a segment. In statistics, for a given random distribution on a single variable and a percentage k, we can obtain a value α such that the interval $[\mu - \alpha\sigma, \mu + \alpha\sigma]$ contains k percent of the population following the given random distribution, where μ is the mean and σ is the standard deviation of the distribution. For example, for a normal distribution and $k = 99.7$, we have $\alpha = 3$.

Given the CNT density variation, the delay of a CNFET circuit is a random variable. Suppose that the mean and the standard deviation of the circuit delay are μ_{ckt} and σ_{ckt}, respectively. Since we want to improve the timing yield, it is equivalent to reducing the sum $\mu_{ckt} + \alpha\sigma_{ckt}$ that corresponds to a given yield of the circuit. Therefore, we can use $\mu_{ckt} + \alpha\sigma_{ckt}$ as a cost function for minimization. For simplicity, we choose $\alpha = 3$ in our experiments.

However, since the mean value μ_{ckt} and the standard deviation σ_{ckt} depend on all the gates in the circuit, they take a large computational effort to calculate. Furthermore, it is hard to optimize $\mu_{ckt} + \alpha\sigma_{ckt}$ directly. Instead, we select *statistically critical segments* and minimize $\mu(d_{seg}) + \alpha\sigma(d_{seg})$ of them, where d_{seg} denotes the delay of a segment. This process effectively improves the statistical performance of the whole circuit. We refer to $\mu(d_{seg}) + \alpha\sigma(d_{seg})$ as the **statistical delay measure for a segment**. Next, we will derive the expressions for $\mu(d_{seg})$ and $\sigma(d_{seg})$.

Assume that there are n gates $G_1, G_2, \ldots, G_n$ along a segment. Each gate drives a stage in the segment. Then, the segment delay d_{seg} is the sum of the stage delays over all the stages, i.e.,

$$d_{seg} = \sum_{i=1}^{n} d_{st,i},$$

where $d_{st,i}$ is calculated by Equation (8).

By Equation (9), the mean of the segment delay is

$$\mu\left(d_{seg}\right) = \mu\left(\sum_{i=1}^{n} d_{st,i}\right) = \sum_{i=1}^{n} \mu\left(d_{st,i}\right) \approx \sum_{i=1}^{n} d_{st,i}^{nom}. \tag{15}$$

The variance of the segment delay is

$$\sigma^2(d_{seg}) = \sigma^2\left(\sum_{i=1}^{n} d_{st,i}\right) = \sum_{i=1}^{n} \sigma^2(d_{st,i}) + 2 \sum_{1 \le i < j \le n} \sigma(d_{st,i})\sigma(d_{st,j})\rho(d_{st,i}, d_{st,j}),$$

where $\rho(X, Y)$ represents the correlation coefficient between two random variables X and Y. By Equation (14), we further have

$$\begin{aligned}\sigma^2(d_{seg}) \approx & \sum_{i=1}^{n} \left(C_{load,i}^{nom}\right)^2 \sigma^2\left(R_{out,max,i}\right) \\ & + 2 \sum_{1 \le i < j \le n} C_{load,i}^{nom} C_{load,j}^{nom} \sigma(R_{out,max,i})\sigma(R_{out,max,j})\rho(d_{st,i}, d_{st,j}).\end{aligned} \tag{16}$$

Since as we showed in Section 4.2, the variation of the output resistance of the driving gate dominates the variation of $d_{st,i}$, we have

$$\rho(d_{st,i}, d_{st,j}) \approx \rho\left(C_{load,i}^{nom} \cdot R_{out,max,i}, C_{load,j}^{nom} \cdot R_{out,max,j}\right) = \rho(R_{out,max,i}, R_{out,max,j}).$$

Due to the asymmetric spatial correlation of the CNFET circuits, we treat $\rho(R_{out,max,i}, R_{out,max,j})$ as 1 if the geometric centers of the gates G_i and G_j are in the same row, and 0 otherwise. Therefore, $\rho(R_{out,max,i}, R_{out,max,j}) = M_{i,j}$, where $M_{i,j}$ is the entry at row i and column j of the correlation matrix M on the gates $G_1, \ldots, G_n$.

To further simplify Equation (16), for a gate G_i, we define $d_i = C_{load,i}^{nom}\sigma\left(R_{out,max,i}\right)$, and the vector $\vec{D} = [d_1, d_2, \ldots, d_n]$.

Then, Equation (16) can be simplified as

$$\sigma^2(d_{seg}) \approx \vec{D} M \vec{D}^T,$$

and the standard deviation of the segment delay is

$$\sigma(d_{seg}) \approx \left(\vec{D}M\vec{D}^T\right)^{\frac{1}{2}}. \tag{17}$$

By Equations (15) and (17), the statistical delay measure for a segment is calculated as

$$\mu(d_{seg}) + \alpha\sigma(d_{seg}) \approx \sum_{i=1}^{n} a_{st,i}^{nom} + \alpha \cdot \left(\vec{D}M\vec{D}^T\right)^{\frac{1}{2}}. \tag{18}$$

Although this analysis is done for the segment, we can do the same analysis for a full path in a circuit and derive a similar way to calculate the statistical delay measure of a path. The proposed statistical delay measure for both segment and path will be used in our optimization flows.

6 THE PROPOSED SEGMENT-BASED GLOBAL PLACEMENT METHOD

In this section, we present our proposed segment-based variation-aware global placement method. One novelty of this method is that it is a segment-based approach, which is different from the conventional path-based or net-based global placement method. This approach first identifies a set of statistically critical segments and then optimizes them. The optimization target is the statistical delay measure, which is composed of the mean and the standard deviation of the segment delay. Although it is not equal to our final target, the timing yield, it has a strong correlation with the timing yield: as we will validate in Section 8.1 through Monte Carlo simulation, with the statistical delay measure decreasing, the circuit timing yield increases. Therefore, we choose the statistical delay measure as our optimization target. We begin by showing the overall flow. Then, we describe three key subroutines in the flow.

6.1 Overall Flow

The overall flow of our proposed variation-aware global placement approach is shown in Algorithm 1. The input is a circuit netlist. After performing an initial placement (Line 1), a preliminary timing-driven force-directed placement is applied to distribute all of the standard cells within the placement region [28] (Line 2). We stop the preliminary placement when the average overlap percentage of the standard cells is below a predefined threshold, which is set as 25% in our experiments. After that, the flow performs an optimization loop (Lines 3–9). At the beginning of each iteration, since the locations of some gates are changed in the previous iteration, the timing information of the gates in the circuit is updated. Then, the statistical static timing analysis (SSTA) is called to obtain the statistical timing information of the CNFET circuit (Line 4). How we perform the SSTA will be described in detail in Section 6.2.

The key idea of the proposed technique is to adjust the locations of the gates on the statistically critical segments to reduce their statistical delay measures calculated by Equation (18). After performing the SSTA, a set of statistically critical segments are selected for optimization (Line 5). The selecting procedure will be described in detail in Section 6.3. For each selected segment, we optimize the location of each gate on it (Line 7). The details of this optimization procedure will be discussed in Section 6.4. After all the selected segments are optimized, the next iteration begins.

The iteration stops when the terminating condition is satisfied. For this method, the terminating criterion will be checked after two iterations are finished. If the metric $\mu_{ckt} + 3\sigma_{ckt}$ of the current iteration is larger than that of the previous iteration, then the placement result of the previous iteration is returned. Here, μ_{ckt} and σ_{ckt} are the mean and the standard deviation of the circuit delay evaluated by SSTA, respectively. Otherwise, if the difference of the metric $\mu_{ckt} + 3\sigma_{ckt}$ between these two successive iterations is less than a threshold ϵ, then the placement result of the

ALGORITHM 1: Procedure of the segment-based variation-aware global placement.

Input: an input netlist.
Output: final global placement result.
do an initial placement;
do a preliminary timing-driven global placement;
repeat
  do statistical static timing analysis (SSTA) and obtain timing information of the circuit;
  choose a set R of statistically critical segments (see Algorithm 2);
  for *each selected statistically critical segments in* R **do**
    optimize the statistical delay measure for this segment by relocating its gates (see Algorithm 3);
  end
until *terminating condition is satisfied*;
return final global placement result;

current iteration is returned. The value ϵ is chosen as 0.02 times the value $\mu_{ckt} + 3\sigma_{ckt}$ of the input placement result.

Note that this proposed method belongs to the global placement phase, because the overlap of standard cells is allowed. To produce a legal layout from the output of our global placement method, a legalization method should be further applied.

6.2 Statistical Static Timing Analysis

Statistical static timing analysis is applied here to obtain the statistical timing information of the CNFET circuit, which is used later to select the statistically critical segments. To perform an accurate SSTA, a method based on non-normal distribution approximation can be used [5]. However, it requires much more computation effort than the SSTA based on normal distribution approximation [3, 4]. Since the SSTA is just used to provide a guideline for selecting the statistically critical segments, we do not require it to be very accurate. Therefore, we approximate the stage delay distribution, which is the basic component used in SSTA, as a normal distribution. This reduces the runtime of SSTA and hence, our optimization procedure. Specifically, we approximate the random variable $d_{st,i}$ as

$$d_{st,i} \approx \mu(d_{st,i}) + \sigma(d_{st,i}) \cdot \Delta x, \tag{19}$$

where Δx is a random variable that follows the standard normal distribution $N(\mu = 0, \sigma = 1)$. Furthermore, by substituting Equations (9) and (14) into Equation (19), we obtain

$$d_{st,i} \approx d_{st,i}^{nom} + C_{load,i}^{nom} \cdot \sigma\left(R_{out,max,i}\right) \cdot \Delta x. \tag{20}$$

We apply Equation (20) to the block-based SSTA method proposed in References [3, 4] to obtain the distributions of the arrival time, the required arrival time, and the slack at each vertex in the timing graph of the CNFET circuit. Specifically, after we obtain the mean and the standard deviation of the arrival time μ_{AT} and σ_{AT} at the terminal vertex by forward traversal, we set the required arrival time at the terminal vertex as $\beta \cdot (\mu_{AT} + 3\sigma_{AT})$ with $\beta = 0.9$ and traverse back to calculate the required arrival time and the slack at each vertex. Then, the probability of the slack being less than zero is computed for each vertex [13]. This probability is referred to as the **violation probability** at the vertex [33], and it will guide us to select the statistically critical segments.

6.3 Selecting Statistically Critical Segments

A key step in the proposed algorithm is to select a set of statistically critical segments to be optimized (Line 5 in Algorithm 1). We describe this step in this section. The procedure is shown in Algorithm 2.

ALGORITHM 2: Procedure for selecting a set of statistically critical segments.

Input: a timing graph after SSTA.
Output: a set of selected statistically critical segments R.
1 $R \Leftarrow \emptyset$;
2 **for** *each statistically critical vertex n* **do**
3 expand n into segments in the original timing graph towards the terminal vertex;
4 add the first reported statistically critical segment into the set R;
5 **end**
6 $Q \Leftarrow$ top q source statistically critical vertices with the highest violation probabilities;
7 **for** *each vertex n in the set Q* **do**
8 expand n into segments in the original timing graph towards the terminal vertex;
9 add the first k reported statistically critical segments into the set R;
10 **end**
11 prune the segments fully contained in other selected segments from the set R;
12 **return** *R;*

After doing the SSTA on the timing graph, we first identify all the vertices with violation probabilities larger than a threshold p_{th}. We call them **statistically critical vertices**. These vertices are very likely to lie on the critical paths and should be optimized to improve the statistical performance of the circuit, such as the timing yield. Our method is to link these vertices into segments, which we call **statistically critical segments**.

To achieve this, we expand each statistically critical vertex into a set of statistically critical segments (Lines 2–5). For each vertex, the expansion procedure (Lines 3) is as follows. It is an iterative process and begins with the specific statistically critical vertex. At each iteration, we always keep a list of top m partial segments with the largest violation probability of the last vertex. At the beginning of each iteration, we first identify the segment in the list with the largest violation probability of its last vertex and delete it from the list. If its last vertex is the terminal vertex of the original timing graph, or it is not statistically critical, then we report this segment; otherwise, we expand this segment by adding to it each fanout vertex of its last vertex to obtain new segments. These new segments are then inserted into the list. The list is reordered based on the violation probabilities of the last vertices of the segments and the top m segments are kept. The expansion procedure terminates when the first statistically critical segment has been reported and the segment is added into the set R (Line 4).

After that, we choose the top q source statistically critical vertices with the highest violation probabilities and form a set Q (Line 6). A **source statistically critical vertex** is a statistically critical vertex with no statistically critical fanins. For each vertex in Q, we expand it into segments with the same expansion procedure described above (Line 8). We select the first k reported segments and add them into the set R (Line 9). Finally, we prune the segments fully contained in the other segments in the set R to refine the set of segments to be optimized (Line 11). In our study, we set $p_{th} = 0.001$, $m = 10$, $q = 5$, and $k = 5$. These parameters are chosen to reduce the runtime while maintaining the optimization quality.

Since the vertices on these segments in the set R have larger violation probabilities than other vertices, they have a larger probability to be on the critical path of the circuit. By optimizing these segments, the timing yield of the circuit will be improved implicitly.

We now analyze the worst-case time complexity of Algorithm 2. We denote the total number of the statistically critical vertices as N_c. We first consider the time complexity of expanding one vertex and reporting any number of statistically critical segments. In each iteration of this procedure, one segment is chosen and expanded. Suppose the segment has t fanouts at its last vertex. Then, t new segments are created. Each is obtained by adding one fanout to the original segment. Each new segment is compared to the segments maintained in the current ordered list of size m. It may or may not replace an existing segment in the list depending on the violation probability of its last vertex. Since the size m of the list is a constant and the number of fanouts t is usually bounded by a constant, we can treat the runtime of each iteration as a constant. Thus, the worst-case time complexity is bounded by the maximal number of iterations in the expansion procedure for a vertex.

Now, we derive the maximal number of iterations. For this purpose, we view the list as a set of m containers, each containing a segment. The container indices are fixed and not affected by the sorting. We use an m-tuple to represent the status of iterations in the expansion procedure. The ith entry in the m-tuple records the number of vertices in the segment in the ith container of the list at the beginning of the current iteration. Since except the last vertex, all the other vertices on a segment in the list are statistically critical, thus, each entry in the m-tuple can be any integer between 0 and $(N_c + 1)$. For the first iteration, the m-tuple has a single 1 and all of its remaining entries are 0. In any later iteration, the segment expansion and replacement step can be abstracted as the following operation to the m-tuple: an arbitrary entry in the m-tuple is chosen and increased by 1; then, this new value replaces a number (which could be zero) of the other entries in the m-tuple. It can be proved that during the entire expansion procedure, the same m-tuple will not appear twice. Therefore, the maximal number of iterations is equal to the total number of different m-tuples. Since each entry of the m-tuple takes an integer value between 0 and $(N_c + 1)$, the maximal number of iterations is $(N_c + 2)^m = O({N_c}^m)$.

Since we need to report the first segment for each of the N_c statistically critical vertices and the first k segments for each vertex in the set Q of size q, the total number of expansion procedures is $N_c + q$. Given that $q \leq N_c$, the total runtime is $O({N_c}^{m+1})$. Although the worst-case time complexity is high, the worst cases rarely happen. The actual runtime is still acceptable.

An experiment was done on the benchmark "PID controller" to show the effectiveness of our method for selecting the statistically critical segments. 14nm CNFET standard cell library was used. Our method was applied to the global placement result and it selected a set of 17 segments. To see the effect, we also generated 2000 samples of the CNFET circuit with random CNT counts. For each circuit sample, we obtained its top one critical path, which determines the circuit delay. For the 2,000 generated sample circuits, we obtained a set S_{path} of 2,000 paths. Then, for each selected segment, we checked whether it appears in each path in the set S_{path}. If at least three gates linking consecutively in a segment are contained in a path in S_{path}, then the segment is considered to appear in that path. As a result, each of the top 14 selected segments appears in more than 95% of the paths in the set S_{path}.This demonstrates the effectiveness of our approach for selecting the statistically critical segment.

6.4 Optimizing Statistical Delay Measure for a Single Segment

Another key step in the proposed algorithm is to optimize the statistical delay measure for a selected segment (Line 7 in Algorithm 1). We describe this step in this section. The detailed procedure is shown in Algorithm 3.

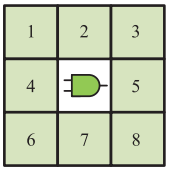

Fig. 6. The eight surrounding grids of a gate, which are searched for the optimal location of the gate.

At the beginning, the statistical delay measure for the input segment sg is calculated by Equation (18) based on the current locations of all the gates on the segment (Line 1). Then, an optimization loop is carried out (Lines 2–17). At the beginning of each iteration, the statistical delay measure for the segment of the previous iteration is recorded (Line 3). During each iteration, the gates along the segment are moved one by one to find their optimal locations that give the minimal statistical delay measure for the entire segment (Lines 4–16). Equation (18) is used to calculate the statistical delay measure for the segment as the gates are moved; it considers the spatial correlation of the gates on the segment. After the statistical delay measure is optimized at the end of each iteration, the improvement of the delay measure over the previous iteration is compared against a threshold th (In our case, th is chosen as 2%). If the improvement is smaller than th, then the entire optimization loop terminates; otherwise, it proceeds to the next iteration (Line 17).

During each iteration, we optimize the location of each gate one by one (Line 4). However, we skip the first and last cells of the segment, since we want to minimize the disturbance to the other part of the circuit caused by relocating the cells along this segment. Furthermore, if a gate on the segment has already been moved, because it belongs to a statistically critical segment that we have handled before, then we also skip this gate. To search for the optimal location for a gate G_i, its surrounding region is divided into eight grids, as shown in Figure 6. Initially, the grid width and height are set as $\frac{1}{3}$ of the width and height, respectively, of the bounding box of the input and output nets of the gate (Lines 5–6). The gate is moved to the center of each of the eight grids and the statistical delay measure for the corresponding segment is obtained (Line 8). The grid gd_j that gives the minimum statistical delay measure among the eight grids is then chosen. If the minimum measure is smaller than the statistical delay measure for the current segment, then the gate is moved to the center of the grid gd_j and the eight grids around the new location are searched again for a better location (Lines 9–11). Otherwise, the grid width and height are reduced by half and the eight surrounding grids with reduced size are searched for a better location (Line 13). The procedure for searching a better location for a gate stops when the grid size is smaller than a threshold h_{min} (Line 7). In this case, the procedure continues toward the next gate. Since this process is greedy, the statistical delay measure for the segment decreases as more gates along the segment are optimized.

We now analyze the worst-case time complexity of Algorithm 3. Assume that the number of gates on a segment is N_s. First, we consider one gate on the segment that is being relocated. Suppose that there are K gates in the circuit in total. A valid assumption is that the placement area is proportional to K. We suppose the area is $A = aK$, where a is a constant. Suppose the initial grid width and height for the gate are w_1 and h_1, respectively. The grid size will be reduced as the while loop proceeds. Suppose the ith grid size has its width as w_i and its height as h_i. We have $w_i = w_1/2^{i-1}$ and $h_i = h_1/2^{i-1}$. Without loss of generality, assume that $w_1 \geq h_1$. Then, the number of different grid sizes is $g_n = \lceil \log_2(h_1/h_{min}) \rceil$, which is a constant. For each grid size, the gate needs to be moved inside the placement region to find its best location. The total number of movements is bounded by the total number of grids of that size in the entire placement region. Thus, for the ith grid size, the total number of movements is bounded by $aK/(w_i h_i)$. Note that once the gate is moved into a new grid, the eight adjacent grids of that new grid will be evaluated.

ALGORITHM 3: Procedure for optimizing the statistical delay measure for a single segment.

Input: a segment sg of N gates.
Output: the segment sg with optimal gate locations to minimize the statistical delay measure.

```
1  calculate the statistical delay measure for sg; denote it as cost;
2  repeat
3      cost_prev ⇐ cost;
4      for each gate G_i other than the first and last ones on the segment sg that has not been moved in other
       segments do
5          get the bounding box of the input and output nets of G_i;
6          initialize the grid width Δx and height Δy to be 1/3 of the width and height of the bounding box,
           respectively;
7          while Δx ≥ h_min and Δy ≥ h_min do
8              move G_i to the center of each of the 8 adjacent grids and record the grid with the
               minimum statistical delay measure; denote the minimum delay measure as cost_min;
9              if cost_min < cost then
10                 move the gate to the grid giving cost_min;
11                 cost ⇐ cost_min;
12             else
13                 Δx ⇐ Δx/2; Δy ⇐ Δy/2;
14             end
15         end
16     end
17 until (cost_prev − cost)/cost_prev < th;
18 return sg;
```

For each evaluation, the amount of computation is dominated by the calculation of the statistical delay measure of the updated segment by Equation (18). This calculation is dominated by a matrix-vector multiplication, where the matrix is of size $N_s \times N_s$ and the vector is of size N_s. Thus, for each movement of the gate, the runtime is $O(N_s^2)$. Consequently, for the ith grid size, the total runtime is $aK/(w_i h_i) \cdot O(N_s^2) = 4^{i-1} \cdot aK/(w_1 h_1) \cdot O(N_s^2)$. Therefore, the total runtime to determine the best location for one gate is $\sum_{i=1}^{g_n} 4^{i-1} \cdot aK/(w_1 h_1) \cdot O(N_s^2)$. Given that w_1, h_1, a, and g_n are constants, the total runtime for one gate is $O(N_s^2 K)$. Assume that the average number of iterations needed to cause the improvement of the statistical delay measure to be less than the threshold th is a constant. Then, the overall runtime to optimize a segment of N_s gates is $O(N_s^3 K)$.

7 THE PROPOSED PATH-BASED GLOBAL PLACEMENT METHOD

Besides the segment-based approach proposed in the previous section, in this section, we propose a path-based global placement method to improve the timing yield of CNFET circuits. It is similar to the segment-based approach except that it optimizes the near critical paths of the circuit. The procedure of the method is shown in Algorithm 4.

In this method, similar as the segment-based method, we first do an initial placement (Line 1), followed by a preliminary timing-driven placement targeting at reducing the nominal delay (Line 2). Then, the procedure enters into an optimization loop with the same termination condition as the segment-based method (Lines 3–10). At the beginning of each iteration, we apply a static timing analysis (STA) to extract the near critical paths of the circuit (Lines 4–5). The paths with nominal delay larger than 0.9 times the nominal circuit delay are selected as the near critical paths. If the number of these selected paths exceeds an upper bound of 1,000, then only the top

1,000 paths are selected. Then, these selected paths are sorted in descending order of their nominal path delay (Line 6), and are treated one after another in this order. For each path, we iteratively relocate the gates on it to minimize the proposed statistic delay measure $\mu + 3\sigma$ for the path by applying Algorithm 3 (Line 8). Note that if a gate has already been moved previously, it is skipped.

ALGORITHM 4: Procedure of the path-based variation-aware global placement.

Input: intermediate global placement result produced by the timing-driven global placement.
Output: final global placement result.

```
1  do an initial placement;
2  do a preliminary timing-driven global placement;
3  repeat
4      do static timing analysis (STA) and obtain timing information of the circuit;
5      choose a set R of near critical paths with nominal path delay larger than 0.9 times the nominal
       circuit delay;
6      sort these near critical paths in descending order of their nominal delays;
7      for each near critical path in the descending order do
8          optimize the statistical delay measure for this path by relocating its gates (see Algorithm 3);
9      end
10 until terminating condition is satisfied;
11 return final global placement result;
```

8 EXPERIMENTAL RESULTS

In this section, we present the experimental results. We first validate the effectiveness of the proposed statistical delay measure. Then, we study the performance of our proposed variation-aware global placement methods.

8.1 Effectiveness of the Statistical Delay Measure

In this work, we propose a statistical delay measure of a segment, which is used to guide the placement of the gates. This measure is intended to have a correlation with the timing yield. More accurately, it should have a correlation with a $p\%$ timing yield margin: the smaller the delay measure is, the smaller the $p\%$ timing yield margin. Here, a $p\%$ timing yield margin is a value such that $p\%$ of the circuits have their delays smaller than this value.

In this section, we validate that the proposed statistical delay measure has a good correlation with a $p\%$ timing yield margin.Note that we do not require this measure to be a close approximation to a $p\%$ timing yield margin. We only require that it has a strong correlation to the $p\%$ timing yield margin. If this is true, then by reducing the delay measure through our proposed placement technique, we can reduce the $p\%$ timing yield margin.

To do the validation, we compared the statistical delay measure of the entire circuit, $\mu_{ckt} + 3\sigma_{ckt}$, with the $p\%$ timing yield margin for two benchmarks "M1-ALU" and "PID controller" [24]. 14nm CNFET standard cell library was used. For each benchmark, we applied the segment-based global placement method and gathered the intermediate global placement results at the end of each optimization iteration. For these two benchmarks, the optimization stops after four and three iterations, respectively, and hence, we have four and three intermediate global placement results, respectively. Together with the input global placement result, we have five and four global placement results for these two benchmarks, respectively. An SSTA was applied to each of these results to evaluate the statistical delay measure $\mu_{ckt} + 3\sigma_{ckt}$ of the entire circuit. For each of these global

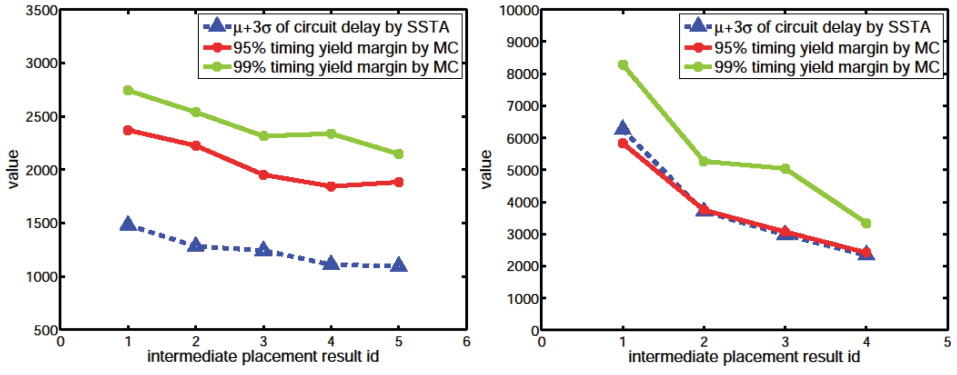

Fig. 7. Statistical circuit delay measure $\mu_{ckt} + 3\sigma_{ckt}$ obtained by SSTA and the 95% and 99% timing yield margins obtained by Monte Carlo simulation (MC) versus the intermediate global placement results during the segment-based global placement for: (a) M1-ALU; (b) PID controller.

placement results, we also applied legalization and then did 2,000 Monte Carlo simulation runs with randomly generated CNT count in each row. By this, we obtained the circuit delay distribution and the 95% and 99% timing yield margins. We plot the statistical delay measure $\mu_{ckt} + 3\sigma_{ckt}$ and the 95% and 99% timing yield margins obtained by Monte Carlo simulation for each of these global placement results. The results are shown in Figure 7.

From the results, we can see that for the benchmark "PID controller," the curve of the statistical circuit delay measure $\mu_{ckt} + 3\sigma_{ckt}$ overlaps with the curve of the 95% timing yield margin. However, in general cases, the statistical delay measure is not close to either the 95% or the 99% timing yield margin. Nevertheless, in general cases, as the statistical circuit delay measure decreases, the 95% and 99% timing yield margins also decrease with the same tendency. The only exception occurs during the convergence period: the timing yield margins may increase a little bit, although the statistical delay measure decreases. However, this is not a significant degradation and can be accepted. Therefore, from this experiment, we justified that although the statistical delay measure is not close to the circuit timing yield margin, it is a good estimator roughly proportional to the timing yield margin. Therefore, we use it to guide the optimization of the circuit timing yield.

8.2 Performance of the Proposed Variation-Aware Global Placement Methods

In this section, we study the performance of our proposed segmented-based and path-based global placement methods.

8.2.1 Experimental Setup. We did experiments with both the 14nm and 10nm CNFET standard cell libraries. The major parameters for both standard cell libraries are listed in Table 2. Seven circuits from OpenCores [24] and a 32-bit multiplier were chosen as our benchmarks, which are listed in Table 3. The sizes of the benchmarks range from 1,096 gates to 16,272 gates. For each benchmark, four experiments were done. Their flows are shown in Figure 8. In the first experiment, the timing-driven force-directed global placement proposed in Reference [28] was applied. This is the baseline method. In the second experiment, the path-healing method adapted from Reference [1] was applied. Note that the original path-healing method is applied to CNFET circuits, where the CNT growth direction is perpendicular to the standard cell rows, while in our study, we assume that the CNT growth direction is parallel to the rows. Furthermore, the original path-healing method does not take into consideration the interconnect delay. To make a fair comparison, we

Table 3. Benchmarks Used in Our Experiment

BENCH	circuit name	#gate	#net
ckt1	M1-ALU	1096	1166
ckt2	32-bit multiplier	2775	2839
ckt3	PID controller	3023	3102
ckt4	cavlc-decoder	3212	3393
ckt5	Gaussian noise generator	5869	5906
ckt6	AES-128	6992	7054
ckt7	M1-CPU	10443	10929
ckt8	M1-CORE	16272	16460

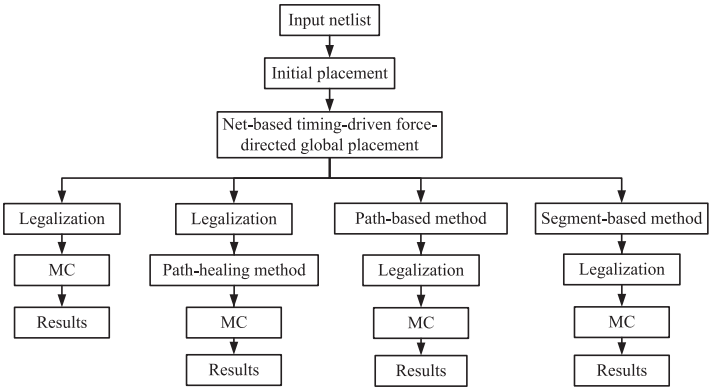

Fig. 8. The flows of four experiments. From left to right are the flows for the baseline method, the path-healing method, the proposed path-based method, and the proposed segment-based method, respectively. MC stands for the Monte Carlo simulation.

modified the path-healing method so that it can handle the same CNFET circuit style as ours and take into account the interconnect delay. In the third and fourth experiments, our proposed path-based and segment-based variation-aware global placement methods were applied, respectively. All the four methods are implemented using *Matlab 2014a*.

For each experiment, the benchmark written in *Verilog* was first synthesized by *Synopsys Design Compiler* to obtain the input circuit netlist. Then, the corresponding global placement algorithm was applied. After the global placement, the legalization approach proposed in Reference [25] was applied to align cells into rows and remove all the overlap among the gates. Then, a Monte Carlo simulation was performed to evaluate the timing statistics of the placement result. For each experiment, 2,000 Monte Carlo simulation runs were carried out.

In each Monte Carlo simulation run, the CNT count for each row of the circuit was generated randomly. The deterministic timing parameters for each standard cell characterized by the method discussed in Section 3.1 were used to obtain the gate delays. HPWL was used as an estimate for the wire-length of each interconnect net. The interconnect delay was evaluated based on Elmore delay model. One pass of the static timing analysis (STA) was applied to the timing graph to evaluate the delay of the circuit. During the STA, the input slew rate for each logic gate is set as the same nominal value; the similar approach is also adopted in Reference [12]. After all the Monte Carlo simulation runs were finished, the different values of circuit delay for all the runs were collected to form a distribution. Then, we obtained the mean, standard deviation, the 95% timing yield margin, and the 99% timing yield margin for this placement.

Table 4. For 14nm Technology Node, Comparison Among the Timing-Driven Force-Directed Global Placement (BASE), the Path-Healing Method (PH), and Our Proposed Path-Based (PATH) and Segment-Based (SEG) Variation-Aware Global Placement Approaches

BENCH	delay μ (ps)				delay σ (ps)				95% TY margin (ps)				99% TY margin (ps)				CPU time (s)		
	BASE	PH	PATH	SEG	BASE	PH	PATH	SEG	BASE	PH	PATH	SEG	BASE	PH	PATH	SEG	PH	PATH	SEG
ckt1	2,098.5	2,317.6	1,751.2	1,690.6	170.6	155.8	169.0	132.6	2,370.0	2,568.2 (+8.4%)	1,937.3 (-18.3%)	1,883.0 (-20.5%)	2,742.2	2,827.1 (+3.1%)	2,470.9 (-9.9%)	2,147.3 (-21.7%)	402	388	398
ckt2	4,612.0	4,387.4	3,176.9	3,490.3	801.1	803.9	575.4	532.2	5,946.7	5,800.4 (-2.5%)	4,304.2 (-27.6%)	4,520.5 (-24.0%)	7,614.2	7,614.2 (0%)	5,720.7 (-24.9%)	5,962.7 (-21.7%)	29	3,517	5,021
ckt3	4,062.0	3,552.4	1,881.5	1,874.3	1,054.7	875.3	388.3	385.1	5,816.8	5,061.2 (-13.0%)	2,380.6 (-59.1%)	2,406.3 (-58.6%)	8,274.5	6,692.2 (-19.1%)	3,214.9 (-61.1%)	3,331.5 (-59.7%)	20	485	351
ckt4	1,891.4	1,884.5	1,332.9	1,335.3	337.3	338.4	250.2	319.5	2,293.4	2,297.8 (+0.2%)	1,683.8 (-26.6%)	1,723.8 (-24.8%)	3,114.6	3,114.6 (0%)	2,354.5 (-24.4%)	2,457.3 (-21.1%)	15	639	522
ckt5	5,858.8	5,780.8	4,014.8	3,786.9	557.3	555.5	640.6	502.5	6,659.1	6,549.6 (-1.6%)	4,921.1 (-26.1%)	4,426.3 (-33.5%)	8,187.1	8,187.1 (0%)	7,078.9 (-13.5%)	6,447.6 (-21.2%)	75	1,263	1,373
ckt6	9,595.2	9,599.7	5,724.4	5,814.6	1,983.4	1,978.7	1,209.8	1,252.6	12,634.7	12,616.9 (-0.1%)	7,730.4 (-38.8%)	7,871.3 (-37.7%)	16,847.8	16,851.7 (+0.02%)	10,710.7 (-36.4%)	10,629.9 (-36.9%)	2,220	5,564	3,209
ckt7	2,283.3	2,275.6	1,663.9	1,501.1	228.9	228.9	371.1	233.2	2,651.0	2,644.6 (-0.2%)	2,245.3 (-15.3%)	1,862.1 (-29.8%)	3,220.4	3,195.1 (-0.8%)	3,125.9 (-2.9%)	2,536.3 (-21.2%)	46	7,335	12,540
ckt8	2,361.7	2,359.4	1,770.1	1,648.7	290.7	290.8	332.0	309.1	2,827.3	2,826.8 (-0.02%)	2,430.5 (-14.0%)	2,256.4 (-20.2%)	3,497.5	3,497.5 (0%)	3,227.9 (-7.7%)	2,846.8 (-18.6%)	56	21,238	26,695

For the 95% and 99% timing yield (TY) margins, the relative difference of the PH/PATH/SEG method over the BASE method is indicated in the parentheses.

8.2.2 The Results for 14nm *Technology Node.* In this section, we present the results for the 14nm CNFET standard cell library. Table 4 shows the delay statistics for all the benchmarks. From the experimental results, we can see that for the mean and the standard deviation of the circuit delay, our proposed segment-based global placement approach gave smaller values than the baseline and the path-healing method for most of the eight benchmarks. The same conclusion holds for our proposed path-based global placement.

Compared to the baseline method, our proposed path-based approach on average improves the 95% and 99% timing yield margins by 28.2% and 22.6% respectively, while our proposed segment-based approach improves the two timing yield margins by 31.1% and 27.8%, respectively. This shows that our proposed variation-aware global placement methods are effective in improving the timing yield of the CNFET circuit. Based on the average improvement percentages, the segment-based approach is better than the path-based approach. For the previous path-healing method, it only improves the 95% and 99% timing yield margins by 1.1% and 2.1% on average, respectively, over the baseline method. Thus, our proposed methods are also much better than the path-healing method. This is due to the following reasons. First, the path-healing method is performed after legalization as a detailed placement method. Therefore, the new locations of the cells are only limited to some legal places. In contrast, our methods are performed before legalization in the global placement phase, which has no restriction on the locations where we can move a cell. Therefore, our methods have a larger search space than the path-healing method. Second, the path-healing method is based on a heuristic golden rule that relocates the standard cells of the same path to different rows. Such a golden rule is just a qualitative guideline. It is less powerful than our proposed statistical delay measure, which is a quantitative measure to guide the optimization. Furthermore, due to its nature of relying on the golden rule, the path-healing method can only run for one iteration, as mentioned in Reference [1]. In contrast, with the quantitative measure, our methods can perform optimization for multiple iterations to continuously improve the timing yield margin.

The last three columns of Table 4 show the CPU time for the three timing-yield optimization methods. Among them, the path-healing method is the fastest, while the proposed path-based and segment-based methods are slower. For some benchmarks, the segment-based method is faster

Table 5. For 10nm Technology Node, Comparison Among the Timing-Driven Force-Directed Global Placement (BASE), the Path-Healing Method (PH), and Our Proposed Path-Based (PATH) and Segment-Based (SEG) Variation-Aware Global Placement Approaches

BENCH	delay μ (ps)				delay σ (ps)				95% TY margin (ps)				99% TY margin (ps)				CPU time (s)		
	BASE	PH	PATH	SEG	BASE	PH	PATH	SEG	BASE	PH	PATH	SEG	BASE	PH	PATH	SEG	PH	PATH	SEG
ckt1	1,887.2	2,121.7	1,643.8	1,663.7	140.8	122.6	158.4	102.8	2,083.7	2,293.2 (+10.1%)	1,796.5 (-13.8%)	1,781.2 (-14.5%)	2,413.5	2,549.5 (+5.6%)	2,385.2 (-1.2%)	1,996.8 (-17.3%)	286	452	364
ckt2	3,649.7	3,467.3	2,682.4	2,824.0	662.3	668.4	530.5	422.8	4,761.3	4,721.1 (-0.8%)	3,625.5 (-23.9%)	3,613.6 (-24.1%)	6,158.5	6,171.4 (+0.2%)	4,986.0 (-19.0%)	4,691.1 (-23.8%)	41	3,966	3,375
ckt3	3,698.8	3,024.5	1,653.6	1,673.6	911.5	703.9	322.0	400.0	5,305.3	4,232.6 (-20.2%)	2,050.8 (-61.3%)	2,195.4 (-58.6%)	7,475.1	5,499.3 (-26.4%)	2,732.0 (-63.5%)	3,731.3 (-50.1%)	25	478	347
ckt4	1,676.6	1,671.7	1,175.1	1,189.9	278.5	279.1	215.9	245.5	2,005.0	2,003.0 (-0.1%)	1,519.1 (-24.2%)	1,517.8 (-24.3%)	2,651.3	2,658.5 (+0.3%)	2,041.8 (-23.0%)	2,020.5 (-23.8%)	17	652	532
ckt5	5,173.8	5,192.2	3,505.2	3,359.3	475.6	472.9	487.5	421.0	5,878.0	5,903.7 (+0.4%)	4,108.8 (-30.1%)	3,861.0 (-34.3%)	7,160.5	7,155.4 (-0.1%)	6,256.5 (-12.6%)	5,683.8 (-20.6%)	77	1,280	1,389
ckt6	8,699.6	8,654.0	5,050.9	5,113.1	1,753.4	1,741.7	1,054.1	1,067.9	11,427.4	11,320.9 (-0.9%)	6,797.6 (-40.5%)	7,032.4 (-38.5%)	14,810.7	14,781.9 (-0.2%)	9,477.7 (-36.0%)	9,029.8 (-39.0%)	2,914	5,933	3,266
ckt7	2,076.2	2,076.3	1,496.2	1,317.1	204.1	203.7	230.4	210.2	2,417.6	2,413.2 (-0.2%)	1,820.3 (-24.7%)	1,642.6 (-32.1%)	2,952.2	2,974.2 (+0.7%)	2,414.3 (-18.2%)	2,206.3 (-25.3%)	43	10,337	11,126
ckt8	2,158.3	2,157.7	1,554.5	1,567.7	238.5	238.5	299.8	252.0	2,607.2	2,574.1 (-1.3%)	2,204.5 (-15.4%)	2,060.5 (-21.0%)	2,989.2	2,990.1 (+0.03%)	2,788.5 (-6.7%)	2,602.3 (-12.9%)	55	27,169	25,737

For the 95% and 99% timing yield (TY) margins, the relative difference of the PH/PATH/SEG method over the BASE method is indicated in the parentheses.

than the path-based method, while for the others, the former is slower. We will study how to further reduce the runtime of our proposed methods in our future work.

8.2.3 The Results for 10nm *Technology Node and Their Comparison with* 14nm *Technology Node.* In this section, we present the results for 10nm CNFET standard cell library and compare the results with those for 14nm CNFET standard cell library.

The delay statistics for the 10nm CNFET standard cell library are shown in Table 5. The numbers shown inside a box in the table are larger than the corresponding numbers in Table 4. For the baseline method, the mean and standard deviation of the circuit delay for 10nm node are smaller than those for 14nm node for all the benchmarks. This is because as the technology node advances, the circuit performance improves. The same conclusion holds for the path-healing method and the proposed path-based method. For the proposed segment-based method, the same conclusion holds for all the benchmarks except *ckt*3.

The comparison of the 95% and 99% timing yield margins between the 10nm and the 14nm technology nodes is shown in Figure 9. In general, for a given method and a given benchmark, both the 95% and 99% timing yield margins are improved as the technology node advances from 14nm to 10nm. For the path-healing method, the average improvement over the baseline on the 95% and 99% timing yield margins is 1.6% and 2.5%, respectively, for the 10nm node. Therefore, same as the 14nm technology node, the path-healing method has very limited improvement on the timing yield margins for the 10nm technology node. For the proposed path-based method, the average improvement over the baseline on the 95% and 99% timing yield margins is 29.2% and 22.5%, respectively, for the 10nm node. The improvement values are both slightly larger than those for the 14nm node. For the proposed segment-based method, the average improvement over the baseline on the 95% and 99% timing yield margins is 30.9% and 26.6%, respectively, for the 10nm node. The improvement values are both slightly smaller than those for the 14nm node. Overall, as the technology advances from 14nm to 10nm, our proposed path-based and segment-based methods can effectively improve the timing yield of CNFET circuits over the baseline method. Same as the 14nm technology node, the segment-based approach is better than the path-based approach for the 10nm technology node.

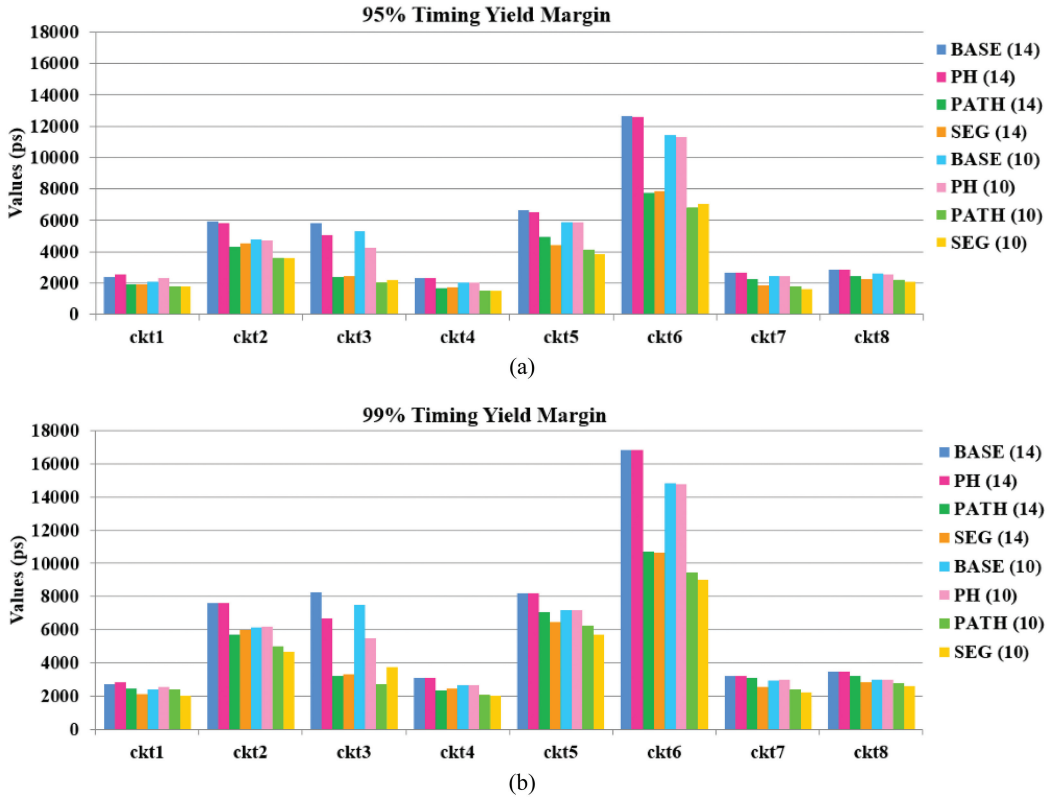

Fig. 9. Histograms of the timing yield margins obtained by Monte Carlo simulation for different global placement methods and technology nodes: (a) 95% timing yield margins; (b) 99% timing yield margins. In the legend, "(14)" and "(10)" denote the 14nm and 10nm technology nodes, respectively.

The last three columns of Table 5 show the CPU time for the three timing-yield optimization methods. The conclusion on CPU time for the 10nm node is same as that for 14nm node.

9 CONCLUSIONS

In this article, we proposed two new variation-aware global placement algorithms for CNFET circuits to improve the circuit timing yield. For the first proposed method, during the global placement, a segment-based optimization technique is applied to place a set of statistically critical segments. It optimizes a statistical delay measure composed of both the mean and the standard deviation of the segment delay. For the second proposed method, the proposed statistical delay measure is optimized for a set of near critical paths. The asymmetric spatial correlation of CNFETs is taken into consideration for both approaches during the optimization process. Experiments showed that our proposed global placement algorithms are effective in improving the timing yields of the CNFET circuits. The segment-based approach is slightly better than the path-based approach.

REFERENCES

[1] Matthias Beste, Saman Kiamehr, and Mehdi B. Tahoori. 2014. Layout-aware delay variation optimization for CNTFET-based circuits. In *Proceedings of the 27th International Conference on VLSI Design and 13th International Conference on Embedded Systems (VLSID'14 and ICES'14)*. IEEE, 393–398. DOI: http://dx.doi.org/10.1109/VLSID.2014.74

[2] Shashikanth Bobba, Jie Zhang, Pierre-Emmanuel Gaillardon, H.-S. Philip Wong, Subhasish Mitra, and Giovanni de Micheli. 2014. System level benchmarking with yield-enhanced standard cell library for carbon nanotube VLSI circuits. *ACM J. Emerg. Technol. Comput. Syst.* 10, 33, 4 (May 2014). DOI: http://dx.doi.org/10.1145/2600073

[3] Hongliang Chang and Sachin S. Sapatnekar. 2003. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In *Proceedings of the 2003 IEEE/ACM International Conference on Computer-aided Design (ICCAD'03)*. IEEE, 621. DOI: http://dx.doi.org/10.1109/ICCAD.2003.129

[4] Hongliang Chang and Sachin S. Sapatnekar. 2005. Statistical timing analysis under spatial correlations. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 24, 9 (Aug. 2005), 1467–1482. DOI: http://dx.doi.org/10.1109/TCAD.2005.850834

[5] Hongliang Chang, Vladimir Zolotov, Sambasivan Narayan, and Chandu Visweswariah. 2005. Parameterized block-based statistical timing analysis with non-Gaussian parameters, nonlinear delay functions. In *Proceedings of the 42nd Annual Design Automation Conference (DAC'05)*. ACM, 71–76. DOI: http://dx.doi.org/10.1145/1065579.1065604

[6] Amit Chowdhary, Karthik Rajagopal, Satish Venkatesan, Tung Cao, Vladimir Tiourin, Yegna Parasuram, and Bill Halpin. 2005. How accurately can we model timing in a placement engine? In *Proceedings of the 42nd Annual Design Automation Conference (DAC'05)*. ACM, 801–806. DOI: http://dx.doi.org/10.1145/1065579.1065792

[7] Jie Deng and H.-S. Philip Wong. 2007. A compact SPICE model for carbon-nanotube field-effect transistors including nonidealities and its application-part II: Full device model and circuit performance benchmarking. *IEEE Trans. Electron. Devices* 54, 12 (Nov. 2007), 3195–3205. DOI: http://dx.doi.org/10.1109/TED.2007.909043

[8] W. C. Elmore. 1948. The transient response of damped linear networks with particular regard to wideband amplifiers. *J. Appl. Phys.* 19, 1 (1948), 55–63. DOI: http://dx.doi.org/10.1063/1.1697872

[9] Behnam Ghavami, Mohsen Raji, and Hossein Pedram. 2011. Timing yield estimation of carbon nanotube-based digital circuits in the presence of nanotube density variation and metallic-nanotubes. In *Proceedings of the 2011 12th International Symposium on Quality Electronic Design (ISQED'11)*. IEEE, 1–8. DOI: http://dx.doi.org/10.1109/ISQED.2011.5770806

[10] Gage Hills, Max Shulaker, Hai Wei, Hong-Yu Chen, H.-S Philip Wong, and Subhasish Mitra. 2014. Robust design and experimental demonstrations of carbon nanotube digital circuits. In *Proceedings of the Custom Integrated Circuits Conference (CICC'14)*. IEEE, 1–4. DOI: http://dx.doi.org/10.1109/CICC.2014.6946036

[11] Gage Hills, Jie Zhang, Charles Mackin, Max Shulaker, Hai Wei, H.-S. Philip Wong, and Subhasish Mitra. 2013. Rapid exploration of processing and design guidelines to overcome carbon nanotube variations. In *Proceedings of the 50th ACM/EDAC/IEEE Design Automation Conference (DAC'13)*. IEEE, 105. DOI: http://dx.doi.org/10.1145/2463209.2488864

[12] Gage Hills, Jie Zhang, Max Marcel Shulaker, Hai Wei, Chi-Shuen Lee, Arjun Balasingam, H.-S. Philip Wong, and Subhasish Mitra. 2015. Rapid co-optimization of processing and circuit design to overcome carbon nanotube variations. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 34, 7 (Mar. 2015), 1082–1095. DOI: http://dx.doi.org/10.1109/TCAD.2015.2415492

[13] Eun Ju Hwang, Wook Kim, and Young Hwan Kim. 2013. Timing yield slack for timing yield-constrained optimization and its application to statistical leakage minimization. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 21, 10 (Oct. 2013), 1783–1796. DOI: http://dx.doi.org/10.1109/TVLSI.2012.2220792

[14] ITRS2013. 2013. International Technology Roadmap for Semiconductors. Retrieved from http://www.itrs2.net/2013-itrs.html.

[15] Albert Lin, Jie Zhang, Nishant Patil, Hai Wei, Subhasish Mitra, and H.-S. Philip Wong. 2010. ACCNT: A metallic-CNT-tolerant design methodology for carbon nanotube VLSI: Analyses and design guidelines. *IEEE Trans. Electron. Devices* 57, 9 (Sept. 2010), 2284–2295. DOI: http://dx.doi.org/10.1109/TED.2010.2053207

[16] Nangate. 2014. Nangate 45nm Open-source Library. Retrieved from http://www.nangate.com/.

[17] Hongsik Park, Ali Afzali, Shu-Jen Han, George S. Tulevski, Aaron D. Franklin, Jerry Tersoff, James B. Hannon, and Wilfried Haensch. 2012. High-density integration of carbon nanotubes via chemical self-assembly. *Nature Nanotechnol.* 7, 12 (Oct. 2012), 787–791. DOI: http://dx.doi.org/10.1038/nnano.2012.189

[18] Nishant Patil, Jie Deng, Albert Lin, H.-S. Philip Wong, and Subhasish Mitra. 2008. Design methods for misaligned and mispositioned carbon-nanotube immune circuits. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 27, 10 (Oct. 2008), 1725–1736. DOI: http://dx.doi.org/10.1109/TCAD.2008.2003278

[19] Nishant Patil, Jie Deng, Subhasish Mitra, and H.-S. Philip Wong. 2009. Circuit-level performance benchmarking and scalability analysis of carbon nanotube transistor circuits. *IEEE Trans. Nanotechnol.* 8, 1 (Jan. 2009), 37–45. DOI: http://dx.doi.org/10.1109/TNANO.2008.2006903

[20] Nishant Patil, Albert Lin, Edward R. Myers, Koungmin Ryu, Alexander Badmaev, Chongwu Zhou, H.-S. Philip Wong, and Subhasish Mitra. 2009. Wafer-scale growth and transfer of aligned single-walled carbon nanotubes. *IEEE Trans. on Nanotechnol.* 8, 4 (Jul. 2009), 498–504. DOI: http://dx.doi.org/10.1109/TNANO.2009.2016562

[21] Nishant Patil, Albert Lin, Jie Zhang, Hai Wei, Kyle Anderson, H.-S. Philip Wong, and Subhasish Mitra. 2009. VMR: VLSI-compatible metallic carbon nanotube removal for imperfection-immune cascaded multi-stage digital logic

circuits using carbon nanotube FETs. In *Proceedings of the International Electron Devices Meeting (IEDM'09)*. IEEE, 1–4. DOI: http://dx.doi.org/10.1109/IEDM.2009.5424295

[22] Nishant Patil, Albert Lin, Jie Zhang, Hai Wei, Kyle Anderson, H.-S. Philip Wong, and Subhasish Mitra. 2010. Scalable carbon nanotube computational and storage circuits immune to metallic and mispositioned carbon nanotubes. *IEEE Trans. Nanotechnol.* 10, 4 (Jul. 2010), 744–750. DOI: http://dx.doi.org/10.1109/TNANO.2010.2076323

[23] Nishant Patil, Albert Lin, Jie Zhang, H.-S. Philip Wong, and Subhasish Mitra. 2009. Digital VLSI logic technology using carbon nanotube FETs: Frequently asked questions. In *Proceedings of the 46th Annual Design Automation Conference (DAC'09)*. ACM, 304–309. DOI: http://dx.doi.org/10.1145/1629911.1629995

[24] OpenCores Project. 2014. OpenCores Projects Benchmarks. Retrieved from http://opencores.org/.

[25] Julia Casarin Puget, Guilherme Flach, Marcelo Johann, and Ricardo Reis. 2015. Jezz: An effective legalization algorithm for minimum displacement. In *Proceedings of the 28th Symposium on Integrated Circuits and Systems Design (SBCCI'15)*. ACM. DOI: http://dx.doi.org/10.1145/2800986.2801013

[26] Ali Arabi M. Shahi and Payman Zarkesh-Ha. 2012. Prediction of gate delay variation for CNFET under CNT density variation. In *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT'12)*. IEEE, 140–145. DOI: http://dx.doi.org/10.1109/DFT.2012.6378214

[27] Max Shulaker, H.-S. Philip Wong, and Subhasish Mitra. 2016. Computing with carbon nanotubes. *IEEE Spectrum* 53, 7 (Jul. 2016), 26–52. DOI: http://dx.doi.org/10.1109/MSPEC.2016.7498155

[28] Peter Spindler, Ulf Schlichtmann, and Frank M. Johannes. 2008. Kraftwerk2-A fast force-directed quadratic placement approach using an accurate net model. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 27 (May 2008), 1398–1411. Issue 8. DOI: http://dx.doi.org/10.1109/TCAD.2008.925783

[29] William Swartz and Carl Sechen. 1995. Timing driven placement for large standard cell circuits. In *Proceedings of the 32nd Annual ACM/IEEE Design Automation (DAC'95)*. ACM, 211–215. DOI: http://dx.doi.org/10.1145/217474.217531

[30] Stanford University. 2015. Stanford VS-CNFET Model. Retrieved from https://nano.stanford.edu/stanford-cnfet2-model.

[31] Chen Wang, Li Jiang, Shiyan Hu, Tianjian Li, Xiaoyao Liang, Naifeng Jing, and Weikang Qian. 2015. Timing-driven placement for carbon nanotube circuits. In *Proceedings of the 28th IEEE International System-on-Chip Conference (SOCC'15)*. IEEE, 362–367. DOI: http://dx.doi.org/10.1109/SOCC.2015.7406983

[32] Neil H. E. Weste and David Money Harris. 2011. *CMOS VLSI Design—A Circuits and Systems Perspective, 4th ed.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

[33] Lin Xie and Azadeh Davoodi. 2011. Bound-based statistically critical path extraction under process variations. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 30, 1 (Dec. 2011), 59–71. DOI: http://dx.doi.org/10.1109/TCAD.2010.2072670

[34] Payman Zarkesh-Ha and Ali Arabi M. Shahi. 2011. Stochastic analysis and design guidelines for CNFETs in gigascale integrated systems. *IEEE Trans. Electron Devices* 58, 2 (Feb. 2011), 530–539. DOI: http://dx.doi.org/10.1109/TED.2010.2092780

[35] Guangyu Zhang, Pengfei Qi, Xinran Wang, Yuerui Lu, Xiaolin Li, Ryan Tu, Sarunya Bangsaruntip, David Mann, Li Zhang, and Hongjie Dai. 2006. Selective etching of metallic carbon nanotubes by gas-phase reaction. *Science* 314, 5801 (2006), 974–977.

[36] Jie Zhang, Shashikanth Bobba, Nishant Patil, Albert Lin, H.-S. Philip Wong, Giovanni De Micheli, and Subhasish Mitra. Carbon nanotube correlation: Promising opportunity for CNFET circuit yield enhancement. In *Proceedings of the 47th ACM/IEEE Design Automation Conference (DAC'10)*. IEEE, 889–892. DOI: http://dx.doi.org/10.1145/1837274.1837497

[37] Jie Zhang, Albert Lin, Nishant Patil, Hai Wei, Lan Wei, H.-S. Philip Wong, and Subhasish Mitra. 2012. Robust digital VLSI using carbon nanotubes. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 31, 4 (Mar. 2012), 453–471. DOI: http://dx.doi.org/10.1109/TCAD.2012.2187527

[38] Jie Zhang, Nishant Patil, Arash Hazeghi, and Subhasish Mitra. 2009. Carbon nanotube circuits in the presence of carbon nanotube density variations. In *Proceedings of the 46th ACM/IEEE Design Automation Conference (DAC'09)*. IEEE, 71–76. DOI: http://dx.doi.org/10.1145/1629911.1629933

[39] Jie Zhang, Nishant P. Patil, Arash Hazeghi, H.-S. Philip Wong, and Subhasish Mitra. 2011. Characterization and design of logic circuits in the presence of carbon nanotube density variations. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 30, 8 (Jul. 2011), 1103–1113. DOI: http://dx.doi.org/10.1109/TCAD.2011.2121010

Received May 2017; revised December 2017; accepted December 2017