Rapid Co-Optimization of Processing and Circuit Design to Overcome Carbon Nanotube Variations

Gage Hills, Jie Zhang, *Member, IEEE*, Max Marcel Shulaker, Hai Wei, *Student Member, IEEE*, Chi-Shuen Lee, Arjun Balasingam, H.-S. Philip Wong, *Fellow, IEEE*, and Subhasish Mitra, *Fellow, IEEE*

***Abstract*—Carbon nanotube field-effect transistors (CNFETs) are promising candidates for building energy-efficient digital systems at highly scaled technology nodes. However, carbon nanotubes (CNTs) are inherently subject to variations that reduce circuit yield, increase susceptibility to noise, and severely degrade their anticipated energy and speed benefits. Joint exploration and optimization of CNT processing options and CNFET circuit design are required to overcome this outstanding challenge. Unfortunately, existing approaches for such exploration and optimization are computationally expensive, and mostly rely on trial-and-error-based *ad hoc* techniques. In this paper, we present a framework that quickly evaluates the impact of CNT variations on circuit delay and noise margin, and systematically explores the large space of CNT processing options to derive optimized CNT processing and CNFET circuit design guidelines. We demonstrate that our framework: 1) runs over 100× faster than existing approaches and 2) accurately identifies the most important CNT processing parameters, together with CNFET circuit design parameters (e.g., for CNFET sizing and standard cell layouts), to minimize the impact of CNT variations on CNFET circuit speed with ≤5% energy cost, while simultaneously meeting circuit-level noise margin and yield constraints.**

***Index Terms*—Carbon nanotube (CNT), CNT variations, delay optimization, design-technology co-optimization.**

I. Introduction

WHILE physical scaling of silicon-based field-effect transistors has improved digital system performance for decades [10], continued device scaling is becoming increasingly challenging [2]. Carbon nanotube (CNT) field-effect transistors (CNFETs) are excellent candidates for continuing to improve both performance and energy efficiency of digital systems [13]. CNFET-based very large-scale integrated (VLSI) digital systems are projected to improve energy-delay product (EDP) by an order of magnitude versus silicon-CMOS [6], [46]. Furthermore, CNFETs provide an exciting opportunity to enable monolithic 3-D integrated circuits [47], leading to additional EDP benefits for CNFET-based digital systems with massive integration of logic and memory [42].

Manuscript received March 1, 2014; revised June 13, 2014; accepted September 22, 2014. Date of publication March 23, 2015; date of current version June 16, 2015. This work was supported in part by the National Science Foundation through the NCN-NEEDS program, contract 1227020-EEC. This paper was recommended by Associate Editor Y. Cao.

G. Hills, J. Zhang, M. M. Shulaker, H. Wei, C.-S. Lee, A. Balasingam, and H.-S. P. Wong are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: ghills@stanford.edu).

S. Mitra is with the Department of Electrical Engineering and the Department of Computer Science, Stanford University, Stanford, CA 94305 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2015.2415492

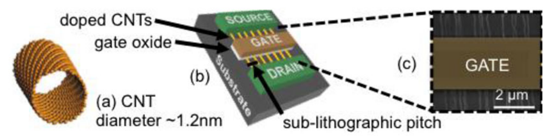

Fig. 1. (a) CNT. (b) Typical CNFET structure. (c) Scanning electron microscopy image of the CNFET channel.

The schematic of a CNFET is shown in Fig. 1. Multiple CNTs compose the transistor channel, whose conductance is modulated by the gate. The gate, source, and drain are defined using traditional photolithography, while the CNT-CNT spacing is determined by the CNT growth [31] and can therefore exceed the minimum lithographic pitch. For high drive current, the target CNT-CNT spacing is 4–5 nm [46].

Despite demonstrations of sub-10 nm channel length CNFETs [13] and stand-alone CNFET circuit elements [5], [7], [11], realization of complex CNFET-based digital systems had been prohibited by substantial imperfections inherent to CNTs: mis-positioned CNTs and metallic CNTs. Mis-positioned CNTs cause stray conducting paths that can lead to incorrect logic functionality, and metallic CNTs (resulting from the imprecise control over CNT properties) result in increased leakage current and can lead to incorrect logic functionality. A unique combination of CNT processing and CNFET circuit design techniques, known as the imperfection-immune paradigm [54], overcomes these challenges in a VLSI-compatible manner to enable the realization of the first CNFET-based digital systems [32], [33], [40], including the first programmable microprocessor built using CNFETs [39]. Two key enablers of these demonstrations are: 1) mis-positioned CNT-immune layout design [30] and 2) VLSI-compatible metallic CNT removal (VMR), which efficiently removes ≥99.99% of metallic CNTs [32], [40].

Unfortunately, process variations specific to CNTs, such as the imprecise control over CNT properties and the nonuniform density of grown CNTs (details in Section II), can lead to significantly reduced circuit yield, increased susceptibility to noise, and large variations in CNFET circuit delays (Section II) [54]. One method to counteract these effects is to upsize all CNFETs. However, such naïve upsizing incurs large energy and delay costs that diminish CNFET technology benefits.

Rather, various CNT process improvement options, when combined with CNFET circuit design, provide an energy-efficient method of overcoming CNT variations. Without such strategies, CNT variations can degrade the potential speed benefits of CNFET circuits by ≥20% at sub-10 nm nodes, even for circuits with upsized CNFETs to achieve ≥99.9% yield (Section II). By leveraging CNT process improvements, together with CNFET circuit design, the overall speed degradation can be limited to ≤5% with ≤5% energy cost while simultaneously meeting circuit-level noise margin and yield constraints [52].

However, co-optimization of CNT technology options and CNFET circuit design parameters using trial-and-error-based search can be prohibitively time-consuming. In this paper, we demonstrate a systematic and VLSI-scalable methodology that selects effective combinations of CNT processing options and CNFET circuit design techniques to overcome CNT variations. Our key contributions are as follows.

1) Techniques to quickly evaluate the impact of CNT variations on circuit yield, susceptibility to noise, delay, and energy. They run >100× faster than previous approaches.
2) A systematic methodology to explore the large space of CNT processing options together with CNFET circuit design parameters (e.g., CNFET sizing and standard cell layouts leveraging CNT correlation, see Section II), to rapidly identify designs that reduce the impact of CNT variations on circuit yield, susceptibility to noise, and delay variations with ≤5% energy cost. This is in sharp contrast to previous trial-and-error-based approaches.
3) Derivation of guidelines for CNT processing and CNFET circuit design parameters at highly scaled technology nodes to overcome CNT variations. We provide guidelines to limit the overall circuit speed degradation to ≤5% with ≤5% energy cost while maintaining ≥99.999% functional circuit yield and ≤0.001% probability of failing to meet circuit-level noise margin requirements (Section IV).

In Section II, we present an overview of CNT variations and their impact on CNFET circuits. Section III describes a methodology to optimize circuit performance in the presence of CNT variations, leveraging a SPICE-compatible CNFET device model to build efficient variation-aware models for the delay, energy, and noise margin of CNFET circuits. Using this methodology, we provide CNT processing and CNFET circuit design guidelines for overcoming CNT variations at the 14, 10, 7, and 5 nm technology nodes (Section IV).

An earlier version of this paper was published in [16]. Here, we present the following additional contributions.

1) Design and analysis of CNFET digital VLSI circuits scaled to the 5 nm node, enabled by a recently developed SPICE-compatible CNFET device model for accurate analysis of sub-10 nm gate length CNFETs [23].
2) A computationally efficient technique to numerically calculate the probability that CNFET circuits fail to meet circuit-level noise margin requirements. This technique can accurately compute such probabilities less than 0.001% (as is desirable for VLSI-scale circuits, details in Section II-C).

In this paper, we make references to [17], which contains additional figures and analysis details. It is available for download at http://www.arxiv.org.

TABLE I
CNT PROCESSING PARAMETERS FOR CNT COUNT VARIATIONS. CNT DENSITY = 250 CNTs/μM FOR ALL ANALYSIS [53]

Proc. Param	Definition	Ideal value	Experimental value
IDC	Index of Dispersion for CNT count $IDC = \sigma_s^2 / \mu_s^2$ μ_s and σ_s^2: mean and variance of the distribution of CNT-CNT spacing [49]	0	0.50 [49]
p_m	Probability that a given CNT is an m-CNT	0%	1%-10% [28], [29]
p_{Rs}	Conditional probability that a CNT is removed, given that it is an s-CNT	0%	4% [40]
p_{Rm}	Conditional probability that a CNT is removed, given that it is an m-CNT	100%	> 99.99% [40]

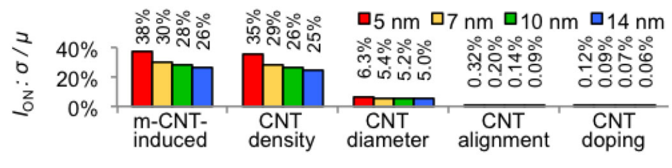

Fig. 2. CNFET I_{ON} variations due to CNT variations (*x*-axis) for a minimum-width CNFET (V_{DD} = 0.50 V, width = half-contacted gate pitch; see [17, Table VI]). $IDC = 0.50$ [49] for CNT density variations, $p_m = 33\%$ [37] and $p_{Rs} = 4\%$ [40] for m-CNT-induced variations. Diameter is normally distributed with μ_d = 1.3 nm and σ_d = 0.1 nm [31]. Alignment and doping distribution details in [54]. To analyze I_{ON} variations attributed to individual sources of CNT variations, all other sources of CNT variations are removed. Additional parameters in [17, Table VI].

II. CNT VARIATIONS

In addition to process variations that exist for silicon-CMOS FETs (e.g., variations in channel length, oxide thickness, and threshold voltage [26]), CNFETs are also subject to CNT-specific variations, including variations in CNT type (semiconducting: s-CNT or metallic: m-CNT) [32], CNT density [49], diameter [34], alignment [30], and doping [9] (details in [17, Sec. VI]). While the on-current (I_{ON}) of a CNFET with only a single CNT as its channel is highly sensitive to CNT diameter variations [34], CNFETs in practical VLSI circuits consist of multiple CNTs to provide sufficient I_{ON}. Thus, the impact of diameter variations is reduced due to statistical averaging (Fig. 2) [35]. Rather, I_{ON} variations are dominated by variations in the CNT count: the number of s-CNTs in a CNFET (after m-CNT removal, e.g., using VMR)[1] [52]. CNT count variations stem from two sources.

1) *CNT Density Variations:* Precise positioning of CNTs is difficult to control; resulting CNT-CNT spacing variations lead to a variable number of CNTs in each CNFET [49].
2) *m-CNT-Induced Variations:* Each CNFET contains a variable number of both s-CNTs and m-CNTs, resulting in CNT count variations even assuming a perfectly selective m-CNT removal technique (i.e., $p_{Rm} = 100\%$, p_{Rs} = 0%: Table I). In addition, m-CNT removal techniques may inadvertently remove a small fraction of s-CNTs, further contributing to CNT count variations [54].

CNT count variations are parameterized by the parameters: Index of Dispersion for CNT count (*IDC*), p_m, p_{Rs}, and p_{Rm} (i.e., the *processing parameters*) defined in Table I. We analyze the impact of CNT count variations on CNFET circuit

[1] Another technique for post-growth m-CNT removal is known as CNT sorting, in which s-CNTs are separated from m-CNTs in a solution [1]. However, CNT sorting techniques have not yet achieved the selectivity required for VLSI-scale digital circuits [50].

modules synthesized from the processor core of OpenSPARC T2, a large multicore chip that closely resembles the commercial Oracle/SUN Niagara 2 system [27]. These OpenSPARC modules consist of ~4 K to >100 K logic gates (Table III) and expose several effects in VLSI-scale circuits (e.g., wire parasitics) that are not visible in small circuit benchmarks. We consider the effects of CNT count variations on the following circuit-level metrics.

1) *Functional Yield:* Due to CNT count variations, there is nonzero probability that a CNFET contains no s-CNTs in its channel, leading to functional failure of the CNFET (i.e., CNT count failure) [51]. The count-limited yield of a CNFET circuit is the probability that no CNFET experiences CNT count failure [51] (Section II-A).
2) *Delay Penalty:* The increase in the 95-percentile-delay (T_{95}: the minimum clock period that the circuit has a 95% probability of meeting) relative to the nominal delay (the critical path delay when there are no variations). Details in Section II-B.
3) *Static Noise Margin (SNM):* A measure of the noise susceptibility of a pair of connected logic gates (Section II-C).
4) *Probability of Noise Margin Violation (PNMV):* The probability that any pair of connected logic gates in a circuit fails to meet SNM_R, a required SNM level (Section II-C).

A. Impact on Circuit Functional Yield

For VLSI CNFET circuits with minimum-width CNFETs, the count-limited yield can be very low (near zero) [51]. An effective method to significantly improve the count-limited yield (≥99.999%) is to perform minimum-width upsizing: upsize all CNFETs that have width (W) less than a specified minimum width (W_{MIN}) to have $W = W_{\text{MIN}}$ [51]. Although minimum-width upsizing effectively improves count-limited yield, it can incur large energy costs if the CNT count failures of all CNFETs are independent [51]. Rather, for CNFET circuits with highly aligned CNTs, the count-limited yield (and the energy cost of minimum-width upsizing, details below) can be significantly improved by leveraging the unique property of CNT correlation: since CNTs are 1-D nanostructures with lengths typically much longer than the CNFET contacted gate-pitch [20], [31], the CNT counts of CNFETs can be uncorrelated or highly correlated depending on the relative physical placement of the CNFET active regions (active region: area of channel which has CNTs) [51]. Special aligned-active layouts can engineer these correlations by aligning the active regions in a library to maximize correlation [17, Fig. 15]. Aligned-active layouts incur minimal area increase (only 4 of 134 cells from the Nangate 45 nm Open Cell Library [25] incur area penalties <14%), and the locations of I/O pins are mostly retained, resulting in negligible impact on intercell routing [51].

To achieve count-limited yield ≥99% for circuits today (which can consist of 100M logic gates), the count-limited yield for each OpenSPARC module (~100K logic gates) should be ≥99.999%. To reach this target, we use a combination of minimum-width upsizing, aligned-active layouts, and CNT process improvements. We first use minimum-width upsizing with aligned-active layouts to achieve count-limited yield ≥99.9% (which is lower than the 99.999% requirement, details below). Then, CNT process improvements (which are required to meet delay penalty and noise margin requirements) further improve the count-limited yield, e.g., to ≥99.999% (details below in steps 1–3). We define ΔE_{FUNC} as the energy cost (in terms of total energy per cycle) of minimum-width upsizing to reach a desired count-limited yield (i.e., functional yield). ΔE_{FUNC} can be ≤2.5% for all the OpenSPARC modules (Fig. 4). It is determined using the design flow in Fig. 3. Steps 1–3 (Fig. 3) are described below.

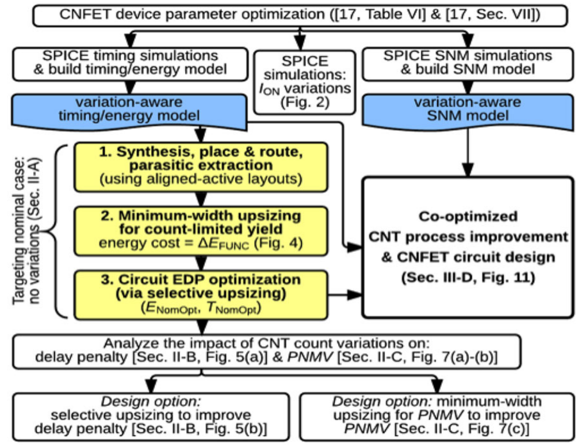

Fig. 3. Full analysis & design methodology. Steps 1–3 (highlighted) are described in this section. Additional details in Section III and [17, Sec. VII].

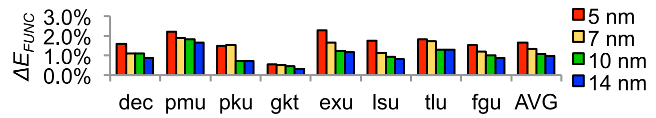

Fig. 4. Energy cost of minimum-width upsizing with aligned-active layouts to achieve ≥99.9% count-limited yield: OpenSPARC modules, $IDC = 0.50$, $p_m = 10\%$, $p_{\text{Rs}} = 4\%$, $p_{\text{Rm}} = 99.99\%$ (count-limited yield improves to ≥99.999% with the processing guidelines in Section IV). Improving delay penalty and *PNMV* can require additional energy costs.

1) *Synthesis, Place and Route, and Parasitic Extraction:* Targeting the nominal case: no variations. Details in [17, Sec. VII].
2) *Minimum-Width Upsizing for Count-Limited Yield:* Determine W_{MIN} to achieve count-limited yield ≥99.9% with aligned-active layouts via the methodology in [51], using experimentally demonstrated values for the processing parameters (Table I, though other values may be chosen). Then perform minimum-width upsizing (the associated energy cost is ΔE_{FUNC}). Note that, this initial count-limited yield target of ≥99.9% is lower than the required ≥99.999% count-limited yield. In Section IV, we show that CNT process improvements (which are required to meet delay penalty and noise margin requirements) further improve the count-limited yield, e.g., to ≥99.999%. If count-limited yield ≥99.999% is not achieved after meeting delay penalty and noise margin requirements, we return to this step and increase W_{MIN} to the width of the next-largest CNFET in our standard cell library (details in [17, Sec. VII]).
3) *Circuit EDP Optimization:* We use the EDP metric to quantify energy efficiency. We perform circuit sizing to minimize circuit EDP using a selective transistor/logic gate upsizing algorithm (i.e., selective upsizing) inspired

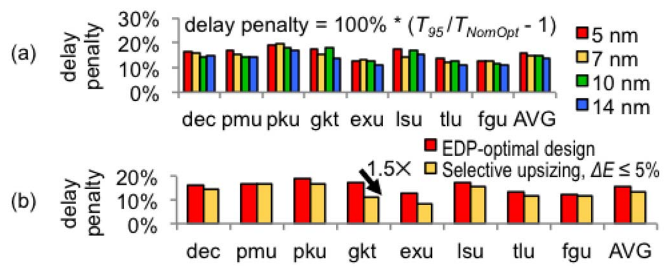

Fig. 5. Delay penalty for the OpenSPARC modules (after steps 1–3 in Fig. 3). $IDC = 0.50$, $p_m = 10\%$, $p_{\mathrm{Rs}} = 4\%$, $p_{\mathrm{Rm}} = 99.99\%$. (a) Delay penalty across technology nodes. (b) Delay penalty improvement due to selective upsizing with $\Delta E \leq 5\%$. For both (a) and (b): count-limited yield ≥99.98% in all cases (it improves to ≥99.999% with the processing guidelines in Section IV).

by [52]: targeting the nominal case, we first sort all standard cells according to their fan-out (fan-out: the ratio of the output load capacitance to the minimum input capacitance on any input), then upsize a parameterized number $k_{\mathrm{SelUpsize}} \geq 0$ of the standard cells with the largest fan-out (see algorithm in [17, Sec. VII-C]). We sweep $k_{\mathrm{SelUpsize}}$ to generate an energy-delay tradeoff curve. We record the nominal energy (E_{Nom}) and the nominal critical path delay (T_{Nom}) for each point on this curve, and then select the point with the minimum EDP [EDP_{NomOpt}, defined in (1)]. This point (for the nominal case) is referred to as the EDP-optimized nominal design point: (E_{NomOpt}, T_{NomOpt}). All delay penalties are relative to this point

$$EDP_{\mathrm{NomOpt}} = E_{\mathrm{NomOpt}} T_{\mathrm{NomOpt}}. \tag{1}$$

While (E_{NomOpt}, T_{NomOpt}) represents an attractive design in the nominal case (since EDP_{NomOpt} is small versus other points on the energy-delay tradeoff curve), this design may have a high delay penalty due to CNT variations (e.g., it can be ≥20% at sub-10 nm nodes: Section II-B).

B. Impact on Circuit Delay Variations

To derive distributions of CNFET circuit delays resulting from CNT count variations, we leverage the methodology described in [52]. This is a Monte Carlo statistical static timing analysis (MC SSTA) approach with two key changes: 1) a variation-aware timing model for CNFET logic gates (built using a CNFET device model [45]) and 2) highly efficient CNT count sampling, based on the unique asymmetric CNT correlation property (Section II-A). This allows us to compute the delay penalty for each OpenSPARC module (after steps 1–3 in Fig. 3) as follows: sample the delay distribution via MC SSTA (using 2000 trials, excluding any trials that have CNT count failure), then extract T_{95} from the delay distribution to calculate the delay penalty [Fig. 5(a)]. Fig. 5(a) illustrates that the delay penalty for the OpenSPARC modules can be ≥20% for EDP-optimized designs with aligned-active layouts at highly scaled technology nodes.

To overcome CNT variations, we target delay penalty ≤5% with total energy per cycle cost $\Delta E \leq 5\%$ [relative to E_{NomOpt}(1)] to maintain ≥90% of the projected EDP benefits of CNFET circuits, even in the presence of CNT variations. To improve delay penalties we leverage the selective upsizing approach described in Section II-A [52]. Fig. 5(b) shows that selective upsizing can reduce delay penalties by 1.5× (e.g., from 17% to 11% for the "gkt" OpenSPARC module); in Fig. 5(b), additional selective upsizing was performed after steps 1–3 in Fig. 3 by increasing $k_{\mathrm{SelUpsize}}$ to minimize the delay penalty subject to $\Delta E \leq 5\%$.

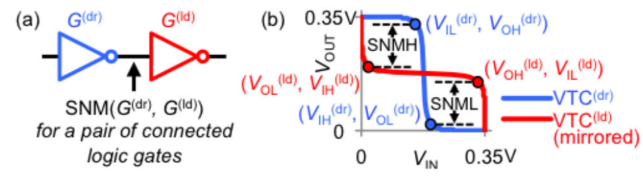

Fig. 6. SNM illustration. (a) Example gate pair. (b) VTC$^{(\mathrm{dr})}$ and mirrored VTC$^{(\mathrm{ld})}$. For each of $G^{(\mathrm{dr})}$ and $G^{(\mathrm{ld})}$: V_{OH}, V_{IH}, V_{IL}, and V_{OL} are taken from the points on the VTC where the slope is −1.

C. Impact on Circuit PNMV

A common metric to quantify the noise susceptibility of a pair of connected logic gates [i.e., a gate pair: ($G^{(\mathrm{dr})}$, $G^{(\mathrm{ld})}$), where $G^{(\mathrm{dr})}$ and $G^{(\mathrm{ld})}$ are the driving and loading logic gates, respectively] is the SNM, which can be quantified as follows [using the gate pair shown in Fig. 6(a) as an example]. Let $G^{(\mathrm{dr})}$ have voltage transfer curve VTC$^{(\mathrm{dr})}$ (voltage transfer curve: V_{OUT} versus V_{IN} in the static case) and let $G^{(\mathrm{ld})}$ have voltage transfer curve VTC$^{(\mathrm{ld})}$. Also, let ($V_{\mathrm{IL}}^{(\mathrm{dr})}$, $V_{\mathrm{OH}}^{(\mathrm{dr})}$) and ($V_{\mathrm{IH}}^{(\mathrm{dr})}$, $V_{\mathrm{OL}}^{(\mathrm{dr})}$) be the points on VTC$^{(\mathrm{dr})}$ where the slope of V_{OUT} versus V_{IN} is −1 [as shown in Fig. 6(b)]. Similarly define ($V_{\mathrm{IL}}^{(\mathrm{ld})}$, $V_{\mathrm{OH}}^{(\mathrm{ld})}$) and ($V_{\mathrm{IH}}^{(\mathrm{ld})}$, $V_{\mathrm{OL}}^{(\mathrm{ld})}$) for VTC$^{(\mathrm{ld})}$ [mirrored in Fig. 6(b)]. Then for the gate pair ($G^{(\mathrm{dr})}$, $G^{(\mathrm{ld})}$), the high SNM (SNMH), the low SNM (SNML), and the SNM are defined in (2)–(4), respectively [48]

$$\mathrm{SNMH}\left(G^{(\mathrm{dr})}, G^{(\mathrm{ld})}\right) = V_{\mathrm{OH}}^{(\mathrm{dr})} - V_{\mathrm{IH}}^{(\mathrm{ld})} \tag{2}$$

$$\mathrm{SNML}\left(G^{(\mathrm{dr})}, G^{(\mathrm{ld})}\right) = V_{\mathrm{IL}}^{(\mathrm{ld})} - V_{\mathrm{OL}}^{(\mathrm{dr})} \tag{3}$$

$$\mathrm{SNM}\left(G^{(\mathrm{dr})}, G^{(\mathrm{ld})}\right) = \min\left(V_{\mathrm{OH}}^{(\mathrm{dr})} - V_{\mathrm{IH}}^{(\mathrm{ld})}, V_{\mathrm{IL}}^{(\mathrm{ld})} - V_{\mathrm{OL}}^{(\mathrm{dr})}\right). \tag{4}$$

SNM($G^{(\mathrm{dr})}$, $G^{(\mathrm{ld})}$) is sensitive to I_{ON} variations [48], and so it is sensitive to CNT count variations. To quantify the impact of SNM variations on circuit noise susceptibility, we use the *PNMV* metric, which is the probability that any gate pair in a circuit fails to meet a required SNM level: SNM_R. SNM_R is a design constraint chosen by the designer and *PNMV* is directly related to SNM_R. As SNM_R increases (tighter SNM requirement) then *PNMV* increases (lower probability of meeting the SNM requirement). Typical values of SNM_R are relative to the supply voltage, V_{DD} (e.g., $SNM_R = V_{\mathrm{DD}}/5$ [48]). *PNMV* is defined in (5), where C is the set of all gate pairs

$$PNMV = 1 - P\left\{\bigcap_{\left(G^{(\mathrm{dr})}, G^{(\mathrm{ld})}\right) \in C} \left(\mathrm{SNM}\left(G^{(\mathrm{dr})}, G^{(\mathrm{ld})}\right) \geq SNM_R\right)\right\}. \tag{5}$$

To solve for *PNMV* due to CNT count variations, we leverage a variation-aware SNM model that can compute SNMH($G^{(\mathrm{dr})}$, $G^{(\mathrm{ld})}$) and SNML($G^{(\mathrm{dr})}$, $G^{(\mathrm{ld})}$) for every gate pair in a circuit, given the CNT counts of each CNFET contained in $G^{(\mathrm{dr})}$ and $G^{(\mathrm{ld})}$ (details in Section III-B1). In Section III-B2, we describe how to combine this variation-aware SNM model and the distributions of CNT count for all CNFETs in the circuit to efficiently calculate *PNMV*.

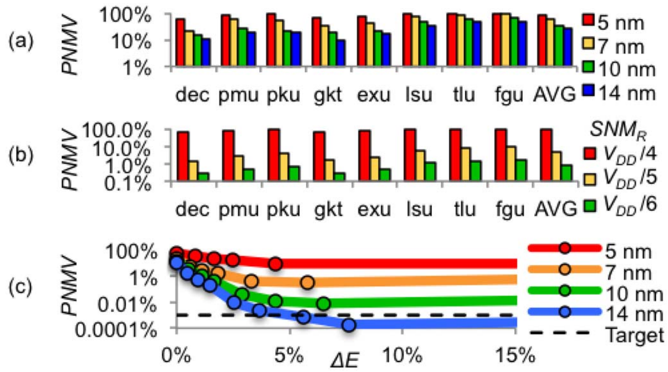

Fig. 7. *PNMV* for the OpenSPARC modules (after steps 1–3 in Fig. 3). $IDC = 0.50$, $p_m = 10\%$, $p_{\mathrm{Rs}} = 4\%$, $p_{\mathrm{Rm}} = 99.99\%$. (a) *PNMV* versus node ($SNM_R = V_{\mathrm{DD}}/4$). (b) *PNMV* versus SNM_R (5 nm node). (c) *PNMV* versus ΔE for additional minimum-width upsizing (in addition to minimum-width upsizing for count-limited yield, Fig. 3): "dec" OpenSPARC module. For (a), (b) and (c): count-limited yield $\geq$99.98% in all cases (it improves to $\geq$99.999% with the processing guidelines in Section IV).

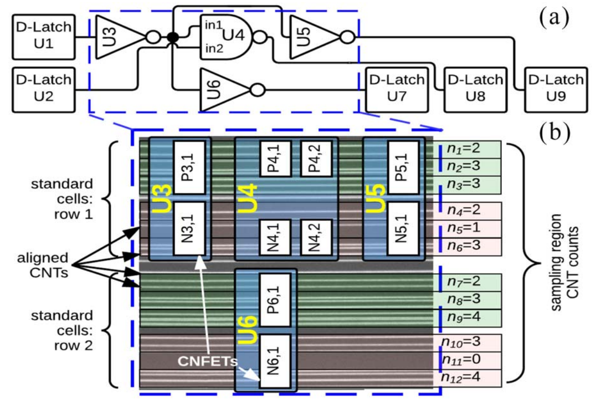

Fig. 8. (a) Subset of logic gates in an example circuit module. (b) Illustration of two rows of standard cells that depicts the relationship between the sampling region CNT counts (e.g., $n_1, n_2, \ldots, n_{12}$) and the CNT counts of each CNFET [53]. For example, the CNT count for CNFET *P*3,1 in inverter *U*3 is $n_1 + n_2 + n_3 = 2 + 3 + 3 = 8$.

Fig. 7(a) and (b) quantify *PNMV* for the OpenSPARC modules (after steps 1–3 in Fig. 3), which can be nearly 100% at the 5 nm node. To achieve $PNMV \leq 1\%$ for circuits today (with ∼100M logic gates), each OpenSPARC module (∼100K logic gates) should have $PNMV \leq 0.001\%$.

Since minimum-width CNFETs are highly sensitive to CNT count variations [52], gate pairs that contain minimum width CNFETs are highly likely to cause SNM violations. Thus, *PNMV* is highly sensitive to minimum-width CNFETs, so further minimum-width upsizing (in addition to minimum-width upsizing for count-limited yield) improves *PNMV* (via statistical averaging) at the cost of energy [Fig. 7(c)]. However, additional minimum-width upsizing may be undesirable as it can require $\Delta E > 5\%$, can increase circuit delay, and is not guaranteed to meet *PNMV* constraints [17, Sec. IX-A].

D. Overcoming CNT Variations

As shown above, CNFET upsizing techniques alone can be insufficient to meet design goals (e.g., delay penalty $\leq$5% and $PNMV \leq 0.001\%$ with $\Delta E \leq 5\%$) [54]. Rather, a combination of CNT processing and CNFET circuit design is required [54], but two key questions must be answered: 1) which processing parameters to improve? 2) By how much?

Without a systematic methodology to evaluate the circuit-level impact of CNT variations, one might blindly pursue difficult CNT processing paths with diminishing returns, while overlooking other processing parameters that enable larger performance gains. For example, much research has focused solely on improving p_m [1]. However, reducing p_m past 1% suffers from diminishing returns and can be insufficient to meet design goals [16], [54] (e.g., in Fig. 16 in [17, Sec. VII]: $p_m = 0.1\%$ does not achieve delay penalty $\leq$5%).

Previously, co-optimization of processing and design has been performed via a trial-and-error-based approach [52]. However, this can be prohibitively time-consuming, potentially requiring months of simulation time (details in Section IV). In Section III, we present a methodology that efficiently selects effective combinations of CNT processing options and CNFET circuit design techniques to overcome CNT variations.

III. Rapid Co-Optimization of Processing & Design

An existing approach to overcome CNT variations is based on brute-force trial-and-error [52]: a designer iterates over many design points (design point: a combination of values for the CNT processing parameters: *IDC*, p_m, p_{Rs}, p_{Rm}, and the CNFET design parameters: e.g., $k_{\mathrm{SelUpsize}}$), analyzing each one until a design point that satisfies a target delay penalty and target *PNMV* with small energy cost is found. Furthermore, this approach utilizes highly accurate yet computationally expensive models to calculate delay penalties and *PNMV*. It suffers from two significant bottlenecks.

1) The time required to calculate delay penalties and *PNMV* limits the number of design points that can be explored.
2) The number of required simulations can be exponential in the number of CNT processing and CNFET design parameters.

Our methodology overcomes these bottlenecks as follows.

1) We estimate delay penalties >100× faster than the previous approach and efficiently calculate $PNMV \leq 0.001\%$, enabling exploration of many more design points while maintaining sufficient accuracy to make correct design decisions (details in Section IV).
2) We use a gradient descent search algorithm, based on delay and *PNMV* sensitivity information with respect to the processing parameters, to systematically guide the exploration of design points (details in Section III-D).

A. Rapid Quantification of Circuit Delay Penalty

To quantify CNFET circuit delay variations, we leverage the probabilistic framework in [52], which is based on an MC SSTA approach with two key enhancements.

1) *Highly Efficient Sampling Method:* It is not trivial to analytically model the effects of CNT correlation at the circuit level. We partition the circuit area in sampling regions, each of which has its own independent CNT count. The CNT count of each CNFET is then the sum of the CNT counts of each sampling region that it overlaps (example shown in Fig. 8) [53].
2) *Variation-Aware Timing Model:* The drive current and parasitic capacitances of CNFETs are modeled as affine functions of the CNT counts in each sampling region [53].

TABLE II
VARIABLES IN THE FULL-CIRCUIT DELAY MODEL

Variable	$\in$	Equation	Description
Q_{MC}	$\mathbb{R}^{m\times n}$	$V_{DD}A_{CLoad}X$	Variable charge per CNT
q_{Exp}	$\mathbb{R}^{m}$	$V_{DD}A_{CLoad}\mathbf{1}$	Expected charge per CNT
q_{Fix}	$\mathbb{R}^{m}$	$V_{DD}b_{CLoad}$	Fixed charge
I_{MC}	$\mathbb{R}^{m\times n}$	$A_{IDrive}X$	Variable current per CNT
i_{Exp}	$\mathbb{R}^{m}$	$A_{IDrive}\mathbf{1}$	Expected current per CNT
i_{Fix}	$\mathbb{R}^{m}$	b_{IDrive}	Fixed current

We incorporate two additional enhancements to improve computation time and for sensitivity analysis of CNFET circuit delay variations versus each processing parameter.

1) *Gaussian Approximation of CNT Count Distributions:* This allows us to factor the variation-aware timing model into two components, one of which does not depend on the processing parameter values and can therefore be precomputed (details below). The CNT count variables are thus elements of the set of real numbers ($\mathbb{R}$) instead of the set of nonnegative integers ($\mathbb{Z}^+$) [49]. The accuracy of this approximation is validated in [17, Sec. VI-B].
2) *Linearized Timing Model for Delay Variations:* We leverage the same timing model as in [53] to compute the maximum path delay of a circuit when no variations are present (nominal delay). Then, we linearize this nonlinear timing model (around the nominal case), and use the resulting linearized timing model to analyze the impact of CNT variations on CNFET circuit delay variations (details in [17, Sec. VIII]). Similar techniques are often used to approximate silicon-CMOS-based circuit delays in early design stages [24]. Unlike in [53], we fix the input slew rate of each logic gate to its nominal value. This allows us to efficiently compute all of the logic gate delays in a circuit simultaneously. These approximations have minimal impact on our design choices (Section IV). We refer to the model in [53] as the nonlinear timing model, and to the model described below as the linearized timing model.

To formulate the delay model for the full circuit, let μ_R and σ_R be the mean and standard deviation of the sampling region CNT count distribution (μ_R and σ_R are functions of the processing parameters shown in Table I). The first step to estimate the delay penalty of a design point is to sample the CNT count for each sampling region and for each MC trial. Each sample is one entry in a matrix $N \in \mathbb{R}^{r\times n}$, where r is the total number of sampling regions and n is the total number of MC trials. We then compute the total capacitive load and drive current for each of the m gates (for each trial) via an affine transformation of the region CNT counts (based on the model in [53]). We express this transformation in matrix form, where $C_{\text{Tot}}, I_{\text{Drive}} \in \mathbb{R}^{m\times n}$

$$C_{\text{Tot}} = A_{\text{CLoad}}N + b_{\text{CLoad}}\mathbf{1}^T \tag{6}$$

$$I_{\text{Drive}} = A_{\text{IDrive}}N + b_{\text{IDrive}}\mathbf{1}^T. \tag{7}$$

Our delay models are fully specified by $A_{\text{CLoad}}, A_{\text{IDrive}} \in \mathbb{R}^{m\times r}$ and column vectors $b_{\text{CLoad}}, b_{\text{IDrive}} \in \mathbb{R}^m$, which contain the coefficients of the affine transformations from the sampling region CNT counts to the CNFET drive currents and parasitic capacitances [53]. Next, we factor out μ_R and σ_R, a crucial step in achieving computational efficiency. We rewrite $N = \mu_R\mathbf{1}\mathbf{1}^T + \sigma_R X$, where each element of $X \in \mathbb{R}^{r\times n}$ is distributed according to a unit Gaussian distribution, allowing (6)-(7) to be written as

$$C_{\text{Tot}} = \sigma_R A_{\text{CLoad}}X + (\mu_R A_{\text{CLoad}}\mathbf{1} + b_{\text{CLoad}})\mathbf{1}^T \tag{8}$$

$$I_{\text{Drive}} = \sigma_R A_{\text{IDrive}}X + (\mu_R A_{\text{IDrive}}\mathbf{1} + b_{\text{IDrive}})\mathbf{1}^T. \tag{9}$$

Note that, $\mathbf{1}$ is a column vector with every element equal to 1, and multiplication of a matrix by a scalar (e.g., μ_R or σ_R) indicates that each element in the matrix is multiplied by that scalar. Any product that does not contain μ_R or σ_R is independent of the processing parameters, and can therefore be precomputed. The dominant computational tasks are the matrix multiplications AX {which are *O(mn)* since A is sparse [12]}. Precomputing such terms (and factoring in the multiplication of C_{Tot} and V_{DD}), yields equivalent expressions for total charge and drive currents that require scalar operations (see Table II for variable definitions)

$$Q_{\text{Tot}} = \sigma_R Q_{\text{MC}} + \left(\mu_R q_{\text{Exp}} + q_{\text{Fix}}\right)\mathbf{1}^T \tag{10}$$

$$I_{\text{Drive}} = \sigma_R I_{\text{MC}} + \left(\mu_R i_{\text{Exp}} + i_{\text{Fix}}\right)\mathbf{1}^T. \tag{11}$$

Precomputing Q_{MC}, q_{Exp}, q_{Fix}, I_{MC}, i_{Exp}, and i_{Fix} (Table II) subsequently allows each logic gate delay to be efficiently computed with only two multiplications, one division, and three additions per trial [only counting operations in (12) that must be computed for each trial]. This includes the addition of $d_{\text{Fix}} \in \mathbb{R}^m$, a vector of fixed delays (e.g., input delays from external circuits). The matrix division in (12) is element-wise

$$D = \frac{\sigma_R Q_{\text{MC}} + \left(\mu_R q_{\text{Exp}} + q_{\text{Fix}}\right)\mathbf{1}^T}{\sigma_R I_{\text{MC}} + \left(\mu_R i_{\text{Exp}} + i_{\text{Fix}}\right)\mathbf{1}^T} + d_{\text{Fix}}\mathbf{1}^T. \tag{12}$$

We then perform static timing analysis (STA) for each MC trial (and for the nominal case), and use the results to estimate T_{95} and the delay penalty. The total circuit energy is computed using a model of the form $E = (1/2)CV^2$ [48]

$$E_{\text{Tot}} = (1/2)V_{\text{DD}}\mathbf{1}^T\left((1/n)\sigma_R Q_{\text{MC}}\mathbf{1} + \mu_R q_{\text{Exp}} + q_{\text{Fix}}\right). \tag{13}$$

B. Rapid Quantification of Circuit PNMV

Our method of analyzing circuit *PNMV* consists of two key components, each of which is described in this section.

1) A variation-aware SNM model, which computes V_{OH}, V_{IH}, V_{IL}, and V_{OL} (these terms are defined in Section II-C) as functions of the CNT counts of the CNFETs within a logic gate. This model can be used to compute SNM for every gate pair in the circuit.
2) A method to numerically calculate low *PNMV* values (e.g., $\leq$0.001%), given the variation-aware SNM model and given a network of cascaded logic gates (e.g., a circuit module after steps 1–3 in Fig. 3). This technique accounts for correlations in CNT count among CNFETs.

1) Variation-Aware Static Noise Margin Model: We refer to V_{OH}, V_{IH}, V_{IL}, and V_{OL} as the VTC parameters, and we model them for each stage of cascaded logic. We distinguish logic stages from standard cells since a standard cell can consist of multiple logic stages (e.g., the standard cell BUF_X1 consists of two cascaded inverters, each of which is one logic stage). For standard cells with multiple logic stages, we model the VTC parameters separately for each logic stage (e.g., we consider the cross-coupled inverters in a D-latch as two separate logic stages). For consistency with the terminology in

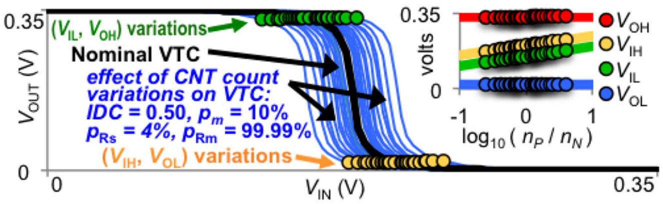

Fig. 9. Variations in the VTC due to CNT count variations (shown for an inverter at the 5 nm node with $V_{\rm DD} = 0.35$ V: e.g., inverter $U3$ in Fig. 8, with $n_P = n_P^{(P3,1)}$ and $n_N = n_N^{(N3,1)}$). Example VTCs: simulated using SPICE by sweeping $V_{\rm IN}$ to obtain $V_{\rm OUT}$. (Inset) VTC parameters versus CNT count. Markers represent extracted values of the VTC parameters, lines represent the SNM model (14) for $T^{(\rm INV\_X1)}$ with $T_{\rm VOH0} = 0.33$, $T_{\rm VOH1} = 0$, $T_{\rm VIH0} = 0.20$, $T_{\rm VIH1} = 0.05$, $T_{\rm VIL0} = 0.15$, $T_{\rm VIL1} = 0.05$, $T_{\rm VOL0} = 0.02$, $T_{\rm VOL1} = 0$.

Section II-C (and without loss of generality), we assume that $G^{(\rm dr)}$ and $G^{(\rm ld)}$ in a gate pair each represent a single logic stage. We also define the state of a logic stage input or output as its logic value (0 or 1). A logic stage input is sensitized if the logic stage output depends on the state of that input (given the logic values of all the other inputs).

For each logic stage input in our standard cell library, we model the VTC parameters for every case in which that input is sensitized (considering all possible combinations of the other inputs). The VTC parameters are functions of the CNT counts of the p- and n-type CNFETs (there is a CNT count variable for each CNFET in the circuit) which: 1) are gated by that input and 2) connect the logic stage output to either $V_{\rm DD}$ or ground through a series of CNFETs in the "on" state (see [17, Sec. IX-B] for an example). We define n_P as the sum of the CNT counts of all such p-type CNFETs. We similarly define n_N for the n-type CNFETs. For example, inverter $U3$ in Fig. 8 consists of a p-type CNFET (labeled "$P3,1$") and an n-type CNFET ("$N3,1$"). The CNT counts of $P3,1$ and $N3,1$ are $n_P^{(P3,1)}$ and $n_N^{(N3,1)}$, respectively. Then $n_P = n_P^{(P3,1)}$ and $n_N = n_N^{(N3,1)}$.

For the NAND2 gate $U4$ in Fig. 8 (as an example of a logic stage with multiple inputs), we separately model the VTC parameters for each input: in1 and in2. Since there are two sets of VTC parameters for the NAND2 gate and only one output, the worst-case values for the output levels $V_{\rm OH}$ and $V_{\rm OL}$ (which are modeled as being independent of the CNT count: details at the end of this section) are selected from the two sets of VTC parameters (i.e., so that SNM is the lowest, see [17, Sec. IX-B] for an example).

Fig. 9 illustrates SPICE simulation data showing that the VTC of a logic stage is sensitive to CNT count variations. For example, the VTCs in Fig. 9 are representative of inverter $U3$ in Fig. 8, with CNFET CNT counts $n_P = n_P^{(P3,1)}$ and $n_N = n_N^{(N3,1)}$; variations in n_P and n_N cause variations in $V_{\rm OH}$, $V_{\rm IH}$, $V_{\rm IL}$, and $V_{\rm OL}$, resulting in SNM variations and larger *PNMV*.

We model the VTC parameters ($V_{\rm OH}$, $V_{\rm IH}$, $V_{\rm IL}$, and $V_{\rm OL}$) as affine functions of $\log_{10}(n_P/n_N)$ (this model is shown for an inverter in Fig. 9), which achieves a root-mean-square (RMS) modeling error $\leq$2.5 mV in all cases (details in [17, Sec. IX-C]). For each case, this affine function is represented by a real-valued matrix $T \in \mathbb{R}^{4\times 2}$

$$\begin{bmatrix} V_{\rm OH} \\ V_{\rm IH} \\ V_{\rm IL} \\ V_{\rm OL} \end{bmatrix} = T \begin{bmatrix} 1 \\ \log_{10}\left(\frac{n_P}{n_N}\right) \end{bmatrix}, T = \begin{bmatrix} T_{\rm VOH0} & T_{\rm VOH1} \\ T_{\rm VIH0} & T_{\rm VIH1} \\ T_{\rm VIL0} & T_{\rm VIL1} \\ T_{\rm VOL0} & T_{\rm VOL1} \end{bmatrix}. \quad (14)$$

To construct the full variation-aware SNM model (consisting of many instances of T in our standard cell library: one for each combination of input states that sensitizes each logic stage input), we perform two steps for each instance of T.

1) Sample the CNT count for each CNFET in the logic stage 2000 times [53] (using the distribution of CNT count, given the CNFET widths and the experimentally demonstrated processing parameter values in Table I), and use SPICE simulations to obtain $V_{\rm OUT}$ versus $V_{\rm IN}$. For each sample, record n_P and n_N and extract $V_{\rm OH}$, $V_{\rm IH}$, $V_{\rm IL}$, and $V_{\rm OL}$ from the VTC in each simulation.
2) Find T via linear regression, given the recorded n_P and n_N and the extracted $V_{\rm OH}$, $V_{\rm IH}$, $V_{\rm IL}$, and $V_{\rm OL}$.

We observed that in all cases, $T_{\rm VOH1} \approx 0$ and $T_{\rm VOL1} \approx 0$ (14), indicating that the CNT count ratio does not strongly affect the output levels of a logic stage.[2] Thus, to simplify our model, we set $T_{\rm VOH1} = 0$ and $T_{\rm VOL1} = 0$, and maintain RMS modeling error $\leq$2.5 mV in all cases (details in [17, Sec. IX-C]). In Fig. 9, the VTC parameters are plotted versus $\log_{10}(n_P/n_N)$.

This variation-aware SNM model is critical for efficiently computing *PNMV* due to CNT count variations, as it relates the VTC parameters to the CNFET CNT counts for each logic stage (14). However, solving (5) for *PNMV* (in Section II-C) is not trivial due to CNT correlation (Section II-A), which causes correlated SNM among gate pairs. In Fig. 8, for example, gate pairs ($U1$, $U3$) and ($U3$, $U5$) have correlated SNM since the CNFETs in $U3$ and $U5$ have correlated CNT counts (they overlap the same sampling regions).

2) Full-Circuit PNMV Model: Here, we demonstrate how the variation-aware SNM model is used to efficiently calculate *PNMV* $\leq$ 0.001% (which is desirable for VLSI-scale circuits: Section II-C) without using an MC-based technique (which would require many trials: e.g., $>10^5$ since 0.001% $= 1/10^5$). There are two key aspects in our framework for computing *PNMV*.

1) *PNMV Formulation:* We formulate *PNMV* as a function of the sampling region CNT count variables (which are independent) to account for the effects of CNT correlation (Section II-A) on SNM.
2) *Solving the PNMV Formulation Efficiently:* We provide a systematic technique to identify a small subset of all SNM constraints in the circuit [i.e., in (5) in Section II-C], referred to as the critical SNM constraints, which are the only SNM constraints that are required to compute *PNMV*. Due to CNT correlation, an SNM violation in a noncritical SNM constraint implies that there must also be an SNM violation in a critical SNM constraint; hence, the noncritical SNM constraint is not required to compute *PNMV*. For the OpenSPARC modules, <1% of all SNM constraints can be critical SNM constraints [17, Table VIII]. Hence, the time to compute *PNMV* (proportional to the number of SNM constraints) can improve by >100×.

The first step to compute *PNMV* is to convert the SNM constraints in (5) into constraints on the CNT counts of each CNFET [using (2), (3), and (5) in Section II-C]. For each gate

[2]The effect of the CNT count ratio on the VTC is similar to that of the "beta ratio" β_P/β_N (a measure of the relative strength of the pull-up and pull-down networks) in silicon-CMOS-based circuits, which does not have a strong effect on the output levels $V_{\rm OH}$ and $V_{\rm OL}$ [48].

pair ($G^{(\mathrm{dr})}$, $G^{(\mathrm{ld})}$) [e.g., ($U3$, $U4$) in Fig. 8], each SNMH constraint has the form in (15) and each SNML constraint has the form in (16) (there can be multiple SNMH and SNML constraints for a single gate pair, details below)

$$\mathrm{SNMH}\left(G^{(\mathrm{dr})}, G^{(\mathrm{ld})}\right) : V_{\mathrm{OH}}^{(\mathrm{dr})} - V_{\mathrm{IH}}^{(\mathrm{ld})} \geq SNM_R \tag{15}$$

$$\mathrm{SNML}\left(G^{(\mathrm{dr})}, G^{(\mathrm{ld})}\right) : V_{\mathrm{IL}}^{(\mathrm{ld})} - V_{\mathrm{OL}}^{(\mathrm{dr})} \geq SNM_R. \tag{16}$$

We then substitute the variation-aware SNM model (14) into these constraints, using $T^{(\mathrm{dr})}$ and $T^{(\mathrm{ld})}$ to represent the SNM model for $G^{(\mathrm{dr})}$ and $G^{(\mathrm{ld})}$, respectively [e.g., $T^{(\mathrm{dr})} = T^{(\mathrm{INV\_X1})}$ and $T^{(\mathrm{ld})} = T^{(\mathrm{NAND2\_X1\text{-}in1})}$ for ($U3$, $U4$) in Fig. 8]

$$T_{\mathrm{VOH0}}^{(\mathrm{dr})} - T_{\mathrm{VIH0}}^{(\mathrm{ld})} - T_{\mathrm{VIH1}}^{(\mathrm{ld})} \log_{10}(n_P/n_N) \geq SNM_R \tag{17}$$

$$T_{\mathrm{VIL0}}^{(\mathrm{ld})} + T_{\mathrm{VIL1}}^{(\mathrm{ld})} \log_{10}(n_P/n_N) - T_{\mathrm{VOL0}}^{(\mathrm{dr})} \geq SNM_R. \tag{18}$$

These constraints are equivalently expressed in matrix form

$$\begin{bmatrix} 1 & \tilde{H}_{1,2} \\ \tilde{H}_{2,1} & 1 \end{bmatrix} \begin{bmatrix} n_P \\ n_N \end{bmatrix} \preceq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{19}$$

$$\tilde{H}_{1,2} = -10^{\left(\left(T_{\mathrm{VOH0}}^{(\mathrm{dr})} - T_{\mathrm{VIH0}}^{(\mathrm{ld})} - SNM_R\right) / T_{\mathrm{VIH1}}^{(\mathrm{ld})}\right)} \tag{20}$$

$$\tilde{H}_{2,1} = -10^{\left(\left(T_{\mathrm{VIL0}}^{(\mathrm{ld})} - T_{\mathrm{VOL0}}^{(\mathrm{dr})} - SNM_R\right) / T_{\mathrm{VIL1}}^{(\mathrm{ld})}\right)}. \tag{21}$$

Note that, the vector inequality in (19) is element-wise (as are all vector inequalities in this section). To account for all SNM constraints in the circuit, let c be the total number of SNM constraints, and let t be the total number of CNFETs (each with its own CNFET CNT count variable, e.g., $n_P^{(P3,1)}$ for CNFET $P3,1$ in Fig. 8). For every gate pair ($G^{(\mathrm{dr})}$, $G^{(\mathrm{ld})}$), there is an SNMH constraint and an SNML constraint for each combination of input states that sensitizes the input to $G^{(\mathrm{ld})}$ that is driven by $G^{(\mathrm{dr})}$. For example, if $G^{(\mathrm{dr})}$ drives an input of $G^{(\mathrm{ld})}$ that can be sensitized by three combinations of input states [e.g., input "A" of an "and-or-invert" logic stage with Boolean function: out = $(\mathrm{A} + (\mathrm{B}^*\mathrm{C}))'$], then there are three SNMH and three SNML constraints for that gate pair (which may constrain different CNT count variables).

The total number of SNM constraints in the circuit is c, and each one imposes a constraint on the CNFET CNT count variables (e.g., $n_P^{(P3,1)}$, $n_N^{(N3,1)}$, $n_P^{(P4,1)}$, etc., in Fig. 8). We can represent these c constraints with a single matrix inequality, by first defining a column vector $s \in \mathbb{R}^t$ that contains the CNT count variables for all the CNFETs in the circuit (e.g., if the entire circuit consisted of the ten CNFETs shown in Fig. 8, then $s = [n_P^{(P3,1)}; n_N^{(N3,1)}; n_P^{(P4,1)}; n_N^{(N4,1)}; n_P^{(P4,2)}; n_N^{(N4,2)}; n_P^{(P5,1)}; n_N^{(N5,1)}; n_P^{(P6,1)}; n_N^{(N6,1)}]$). Then by using all instances of T in the SNM model (14), we formulate each SNM constraint as a constraint on the vector s, using the same procedure as above to convert (15)-(16) into constraints on the CNT counts in (19) (example in [17, Sec. IX-D]). We express these constraints using a matrix $H \in \mathbb{R}^{c \times t}$, such that satisfying (22) is equivalent to satisfying all SNM constraints in the circuit. [17, Table IX] summarizes all variables in this section.

$$Hs \preceq \mathbf{0}. \tag{22}$$

Note that, $\mathbf{0}$ is a column vector with element entry equal to 0. See [17, Sec. IX-D] for the formulation of (22) for the example circuit shown in Fig. 8. Since all SNM constraints in the circuit are represented in (22) (each of c rows in H represents a single SNM constraint), $PNMV$ (5) is the probability that (22) is violated (i.e., $PNMV = 1 - P\{Hs \preceq \mathbf{0}\}$). However, solving for $PNMV$ using (22) is not trivial since the CNT count variables (i.e., the elements of s) can be highly correlated due to CNT correlation (Section II-A). For example, in Fig. 8, the active regions of CNFETs $P3,1$ and $P5,1$ are aligned, so their CNT counts ($n_P^{(P3,1)}$ and $n_P^{(P5,1)}$) are correlated. Thus, the SNM constraints on $n_P^{(P3,1)}$ and the SNM constraints on $n_P^{(P5,1)}$ are dependent.

We can reformulate $PNMV$ to efficiently account for CNT correlation by transforming the constraints in (22) (that constrain the CNFET CNT count variables, which are dependent) into constraints on the sampling region CNT count variables (which are independent). To do so, we first define a column vector $n \in \mathbb{R}^r$ that contains the CNT count variables for all the sampling regions (e.g., in Fig. 8, $n = [n_1; n_2; n_3; n_4; n_5; n_6; n_7; n_8; n_9; n_{10}; n_{11}; n_{12}; \ldots]$). To formulate (22) in terms of vector n instead of vector s (s: the CNFET CNT count variables), the relationship between n and s is required. We express this relationship as a linear transformation represented by a matrix $B \in \{0, 1\}^{t \times r}$ (details below) such that

$$s = Bn. \tag{23}$$

There is one row in B for each CNFET in the circuit, and one column for each sampling region. To determine B: if CNFET i (of t total CNFETs) overlaps sampling region j (of r total sampling regions), then the value of B in row i, column j is 1 (i.e., $B_{i,j} = 1$); otherwise, $B_{i,j} = 0$ (as an example, B for the circuit in Fig. 8 is shown in [17, Sec. IX-D]). Then by substituting (23) into (22), the SNM constraints can be expressed in terms of the region CNT count variables (instead of the CNFET CNT count variables), using a matrix $K \in \mathbb{R}^{c \times r}$

$$K = HB \tag{24}$$

$$Kn \preceq \mathbf{0}. \tag{25}$$

All SNM constraints in the circuit are represented in (25) [just as in (22)], so (25) can also be used to determine $PNMV$ (i.e., $PNMV = 1 - P\{Kn \preceq \mathbf{0}\}$). The advantage of using (25) instead of (22) is that all the variables in n (the vector of sampling region CNT counts) are independent (unlike the correlated variables in s: the vector of CNFET CNT counts). For example, (25) can be used to estimate $PNMV$ via an MC-based approach: for each trial, sample all elements of n (from the distribution of sampling region CNT count) and evaluate Kn. Then estimate $PNMV$ as the fraction of trials that violate (25).

However, evaluating every SNM constraint in (25) is unnecessary since many of them are noncritical SNM constraints (as described above, any SNM constraint that cannot be uniquely violated without simultaneously violating another SNM constraint is not required to compute $PNMV$). See [17, Sec. IX-E] for a detailed description, including examples, of how to systematically identify and eliminate all noncritical SNM constraints.

Eliminating these noncritical SNM constraints is crucial to improve computational efficiency, as they can account for $\geq$99% of all SNM constraints in the circuit (e.g., for the OpenSPARC modules [17, Table VIII]). Since each row of K (25) represents an SNM constraint, we can remove the rows in K that correspond to noncritical SNM constraints to form $\tilde{K} \in \mathbb{R}^{p \times r}$, where p is the number of critical SNM constraints

$$\tilde{K}n \preceq \mathbf{0}. \tag{26}$$

To further improve computational efficiency, we then factor out μ_R and σ_R from the sampling region CNT count variables n in (26) (just as we did for the full-circuit delay model in Section III-A), allowing us to quickly recompute *PNMV* after updating the processing parameter values (details in Section III-D). We rewrite $n = \mu_R \mathbf{1} + \sigma_R x$, where each entry of $x \in \mathbb{R}^r$ is distributed according to a unit Gaussian distribution; then (26) becomes

$$\tilde{K}x \preceq (\mu_R/\sigma_R)b \quad (27)$$

$$b = -\tilde{K}\mathbf{1}. \quad (28)$$

In (27), b is a p-dimensional vector of constants and the matrix-vector product $\tilde{K}x$ is a p-dimensional vector of Gaussian random variables with covariance matrix $C \in \mathbb{R}^{p \times p}$

$$C = \tilde{K}\tilde{K}^T. \quad (29)$$

That is, $\tilde{K}x$ is distributed according to a multivariate normal (MVN) distribution with covariance matrix C; thus, $PNMV = 1 - P\{\tilde{K}x \preceq (\mu_R/\sigma_R)b\}$ can be solved numerically using existing software packages for computing MVN probabilities (see [14]). In particular, consider the cumulative distribution function (CDF) of the MVN-distributed matrix-vector product $\tilde{K}x$ (i.e., the MVNCDF); the probability that all SNM constraints are satisfied (i.e., $1 - PNMV$) is equal to the value of the MVNCDF at the p-dimensional point $(\mu_R/\sigma_R)b$

$$PNMV = 1 - P\left\{\tilde{K}x \preceq (\mu_R/\sigma_R)b\right\} \quad (30)$$

$$PNMV = 1 - \mathrm{MVNCDF}(C, (\mu_R/\sigma_R)b). \quad (31)$$

In [17, Sec. IX-F], we describe how to efficiently solve (31) by leveraging the property that many terms in the covariance matrix, C, are 0; e.g., for the OpenSPARC modules, *PNMV* can be computed in less than 10 seconds using a single 2.93 GHz processor core. In [17, Fig. 21], we validate the accuracy of (31) against MC simulations. For the MC approach, we first sample the vector of sampling region CNT counts [n in (25)] for each trial. Then we estimate *PNMV* as the fraction of samples that violate (25).

C. Circuit Performance Sensitivity to Processing Parameters

Our goal is to achieve small delay penalties and *PNMV* with small ΔE. We quantify the tradeoff between total circuit energy [E_{Tot} in (13)] and delay penalty using EDP_{95}: defined in (32). We also define the energy-*PNMV*-product (*ENP*) metric to quantify the tradeoff between E_{Tot} and *PNMV*

$$EDP_{95} = E_{\mathrm{Tot}} T_{95} \quad (32)$$

$$ENP = E_{\mathrm{Tot}} PNMV. \quad (33)$$

While rapid computation of circuit delay penalty and *PNMV* overcomes the computation time bottleneck of analyzing a single design point, we still require a method for intelligently exploring the large space of CNT processing options. In general, a common measure of the sensitivity of an objective function (e.g., EDP_{95} or *ENP*) with respect to each of its input variables (e.g., the processing parameters) is its gradient. The EDP_{95} and *ENP* gradients are defined in (34)–(36) and are used to guide the exploration of processing options to improve delay penalties and *PNMV* (Section III-D). Fig. 10 illustrates a flowchart of the steps used to compute (32)–(36). The gradients (34)–(36) are computed as

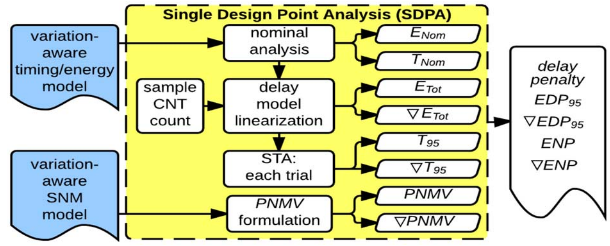

Fig. 10. SDPA to calculate the delay penalty, EDP_{95}, ∇EDP_{95}, *ENP*, and ∇ENP of a single design point.

described [17, Sec. X-A].

$$\nabla E_{\mathrm{Tot}} = \left[\frac{\partial E_{\mathrm{Tot}}}{\partial IDC}; \frac{\partial E_{\mathrm{Tot}}}{\partial p_m}; \frac{\partial E_{\mathrm{Tot}}}{\partial p_{\mathrm{Rs}}}\right]$$

$$\nabla T_{95} = \left[\frac{\partial T_{95}}{\partial IDC}; \frac{\partial T_{95}}{\partial p_m}; \frac{\partial T_{95}}{\partial p_{\mathrm{Rs}}}\right] \quad (34)$$

$$\nabla PNMV = \left[\frac{\partial PNMV}{\partial IDC}; \frac{\partial PNMV}{\partial p_m}; \frac{\partial PNMV}{\partial p_{\mathrm{Rs}}}\right]$$

$$\nabla EDP_{95} = \nabla E_{\mathrm{Tot}} T_{95} + E_{\mathrm{Tot}} \nabla T_{95} \quad (35)$$

$$\nabla ENP = \nabla E_{\mathrm{Tot}} PNMV + E_{\mathrm{Tot}} \nabla PNMV. \quad (36)$$

D. Guided Exploration to Overcome CNT Variations

To overcome the bottleneck of trial-and-error-based search (i.e., iterating over many combinations of values for the processing parameters and design parameters defined in Section II: *IDC*, p_m, p_{Rs}, p_{Rm}, $k_{\mathrm{SelUpsize}}$), we use a gradient descent-based strategy to systematically guide the improvement of EDP_{95} and *ENP* in the presence of CNT variations (while gradient descent strategies can converge to local rather than global optima, in [17, Sec. X-C] we discuss techniques to reduce the impact of local optima during gradient descent in our methodology).

For any design point, we can use single design point analysis (SDPA: Fig. 10) to determine the sensitivity of each circuit performance metric (e.g., EDP_{95} or *ENP*) to each processing parameter by computing its gradient (e.g., ∇EDP_{95} or ∇ENP). These gradients can then be used to identify which processing parameters should be improved (and by how much) to efficiently improve the circuit performance metrics. For example, consider EDP_{95}: $\partial EDP_{95}/\partial IDC$, $\partial EDP_{95}/\partial p_m$, and $\partial EDP_{95}/\partial p_{\mathrm{Rs}}$ indicate how sensitive EDP_{95} is to improvements in *IDC*, p_m, and p_{Rs}. Thus, to effectively improve EDP_{95}, we can update each processing parameter by an amount proportional to its corresponding value in ∇EDP_{95} (e.g., *IDC* corresponds to $\partial EDP_{95}/\partial IDC$). We refer to each such update as a gradient descent step (details in [17, Sec. X-B]).

Before describing the full gradient descent methodology, we define the initial design point as the design point after EDP optimization in the nominal case (i.e., after steps 1–3 in Fig. 3). Also, we define the initial processing parameter values as the processing parameter values of the initial design point (e.g., $IDC = 0.50$, $p_m = 10\%$, $p_{\mathrm{Rs}} = 4\%$, $p_{\mathrm{Rm}} = 99.99\%$: Table I). Starting from the initial design point, we first perform selective upsizing (as described in Section II-A), incrementally increasing $k_{\mathrm{SelUpsize}}$ (the number of standard cells to upsize) to generate a set of design points referred to as the initial energy-delay tradeoff curve (the values of $k_{\mathrm{SelUpsize}}$ are chosen so that each increase in $k_{\mathrm{SelUpsize}}$ increases ΔE by $\sim$1%–2% to identify multiple design points with $\Delta E \leq 5\%$).

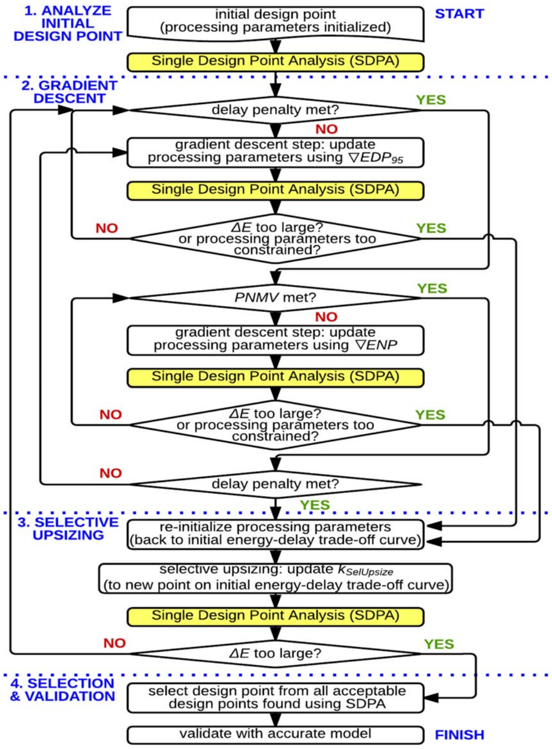

Fig. 11. Gradient descent-based methodology to meet delay penalty, *PNMV*, and ΔE requirements. SDPA details in Fig. 10.

The full methodology, illustrated in Fig. 11, combines selective upsizing and gradient descent to overcome the impact of CNT count variations on delay penalty and *PNMV*. After generating the initial energy-delay tradeoff curve via selective upsizing, our goal is to identify *multiple* design points that meet both a delay penalty constraint (e.g., delay penalty $\leq 5\%$) and a *PNMV* constraint (e.g., $PNMV \leq 0.001\%$) with minimal energy cost (e.g., $\Delta E \leq 5\%$). Such design points that simultaneously satisfy all these design goals are referred to as acceptable design points. Consequently, this is a feasibility problem in which we search for design points that meet two constraints, and we solve it using a variation of an alternating projections (AP) algorithm [3]. A typical AP algorithm iteratively projects a point onto multiple constraints until all are satisfied. In our methodology, we use gradient descent instead of projection; the full methodology (Fig. 11) is described below (example in Fig. 12).

1) *Analyze the Initial Design Point:* Perform SDPA (Fig. 10) on the initial design point [with initial processing parameter values and $k_{\mathrm{SelUpsize}}$ set to minimize EDP_{NomOpt} (1): i.e., after steps 1–3 in Fig. 3].
2) *Gradient Descent:* Alternate between: 1) performing gradient descent steps using ∇EDP_{95} until the delay penalty constraint is satisfied and 2) performing gradient descent steps using ∇ENP until the *PNMV* constraint is satisfied. This procedure continues until either: a) both constraints are satisfied simultaneously (i.e., an acceptable design point is found) or b) ΔE is too large or the processing parameters are too constrained (e.g., a design point with $\Delta E > 5\%$ is reached, or the required processing parameter values may be difficult to achieve experimentally: both are design choices).
3) *Selective Upsizing:* Reinitialize the processing parameters to their initial values (thus returning to the initial energy-delay tradeoff curve) and then perform selective upsizing (by increasing $k_{\mathrm{SelUpsize}}$) to move to the next point on the initial energy-delay tradeoff curve. If ΔE from selective upsizing is too large (e.g., $\Delta E > 5\%$), then proceed to step 4 (below). Otherwise, loop back to step 2 (gradient descent) to search for an additional acceptable design point.
4) *Design Point Selection and Validation:* Select a single design point from all acceptable design points identified using gradient descent. For example, the designer can select the acceptable design point with the minimum EDP_{95} or with the most relaxed processing requirements (a design choice). Finally, highly accurate models (e.g., the nonlinear timing model) can be used to validate the selected design point (if all constraints are not satisfied during validation, then perform additional gradient descent steps until they are satisfied).

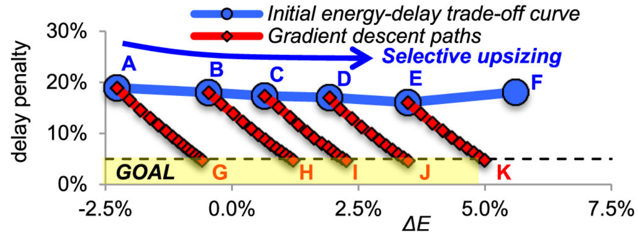

Fig. 12. Gradient descent methodology (Fig. 11) to achieve delay penalty $\leq 5\%$, $PNMV \leq 0.001\%$ (for $SNM_R = V_{\mathrm{DD}}/6$), $\Delta E \leq 5\%$ (5 nm "pku" OpenSPARC module). Gradient descent paths descend from the initial energy-delay tradeoff curve ($IDC = 0.50, p_m = 10\%$, $p_{\mathrm{Rs}} = 4\%$, $p_{\mathrm{Rm}} = 99.99\%$). The point (delay penalty, ΔE) = (0%, 0%) represents the EDP-optimized nominal design point (Section II-A, Fig. 3). $\Delta E < 0$ for point A since E_{Tot} depends on the number of s-CNTs after m-CNT removal (i.e., the CNT count variables), as shown in (13) in Section III-A; due to CNT count variations (e.g., resulting from $p_m > 0\%$, $p_{\mathrm{Rs}} > 0\%$), the total number of s-CNTs in all CNFETs can be reduced versus the nominal case (no variations).

Fig. 12 illustrates an example of the gradient descent-based methodology (Fig. 11) to meet delay penalty $\leq 5\%$ and *PNMV* $\leq 0.001\%$ with $\Delta E \leq 5\%$. Starting from point A (the initial design point), we perform selective upsizing to generate the initial energy-delay tradeoff curve (as described earlier in this section) represented by points A–F. Then, using the methodology in Fig. 11, we perform gradient descent (starting from the initial design point: point A) until delay penalty $\leq 5\%$ and $PNMV \leq 0.001\%$ (at point G: an acceptable design point). Next, the processing parameters are reinitialized and then selective upsizing brings us to point B on the initial energy-delay tradeoff curve. Again, gradient descent is performed to identify another acceptable design point (point H). This process repeats until we reach point F on the initial energy-delay tradeoff curve, which has $\Delta E > 5\%$, concluding the search for acceptable design points.

In Fig. 12, gradient descent has identified *multiple* acceptable design points with varying ΔE and processing requirements. Furthermore, alternative sets of acceptable design points can be identified by adjusting the gradient descent step procedure: e.g., if *IDC* is difficult to improve (i.e., it is difficult to control CNT density variations experimentally), then the gradient descent step can be weighted toward larger updates in p_m or p_{Rs}, or can be forced never to update *IDC* past a predetermined hard-limit. These constraints can be provided as inputs, and are features of this flexible framework.

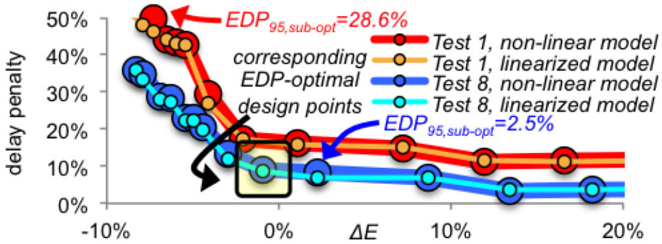

Fig. 13. Delay penalty vs. ΔE (5 nm "gkt" OpenSPARC module). Large markers: values computed using the nonlinear model; small markers: linearized model. Processing parameter values are in [17, Table VII] (tests 1 & 8). Both models identify that the same design point has the minimum EDP_{95}, thus $EDP_{95,\text{sub-opt}} = 0\%$. If the models had selected different design points [e.g., if the linearized model had selected the labeled point at (2.3%, 8.2%) with $EDP_{95,\text{sub-opt}} = 2.5\%$] then $EDP_{95,\text{sub-opt}} \geq 0\%$. The point (delay penalty, ΔE) = (0%, 0%) represents the EDP-optimized nominal design point (Section II-A, Fig. 3). Design points with $\Delta E < 0$ are for designs with smaller CNFET widths.

IV. Results

We present two sets of results to demonstrate that we have overcome the bottlenecks of brute-force trial-and-error-based approaches. The first set of results (Section IV-A) demonstrates that we can analyze a set of design points $>100\times$ faster than before, while maintaining sufficient accuracy to make correct design decisions. The second set of results (Section IV-B) demonstrates the ability of this gradient descent algorithm to identify multiple processing options to meet design goals (e.g., delay penalty $\leq 5\%$ and $PNMV \leq 0.001\%$ with $\Delta E \leq 5\%$) without exhaustive search. Using the results from gradient descent, we provide practical processing guidelines for each node and for multiple values of V_{DD} such that: even in the presence of CNT variations, CNFET circuits can maintain $\geq 90\%$ of the projected EDP benefits of nominal CNFET circuits.

A. Linearized Timing Model Validation

Here, we validate the speed and accuracy of the linearized timing model to analyze circuit delay variations. We first choose a set of design points that would typically be chosen by a designer seeking to optimize EDP_{95} using a brute-force-search-based approach: we use the design points in [52] as a reference. We analyze 112 design points including all combinations of: eight unique sets of processing parameter values (the same as in [52]: see [17, Table VII]), and 14 different $k_{\text{SelUpsize}}$ values (each increase in $k_{\text{SelUpsize}}$ increases ΔE by ~1%–5%, e.g., in Fig. 13; higher resolution requires more computation time).

After choosing the set of design points, we use the nonlinear timing model to compute EDP_{95} of every design point and then select the design point with the best (minimum) EDP_{95}. We also record the total required computation time. We then perform the same procedure using the linearized timing model. We evaluate: 1) the total computation time for each model and 2) the degradation (increase) in EDP_{95} due to using the linearized timing model. To quantify this degradation, we use the EDP_{95} sub-optimality metric defined in (37); it is computed using only the nonlinear timing model to compare: EDP_{95} of the design point selected by each model

$$EDP_{95,\text{ sub-opt}} = \frac{EDP_{95,\text{ Measured By NonLinear Model}}^{(\text{Selected By Linearized Model})}}{EDP_{95,\text{ Measured By NonLinear Model}}^{(\text{Selected By NonLinear Model})}} - 1. \quad (37)$$

TABLE III
EDP_{95} Sub-Optimality and Computation Time (Measured on a Single 2.93 GHz Processor With No Parallelization). Logic Gate Count Is Taken From the Synthesized Netlist at the 5 nm Node (With $V_{\text{DD}} = 0.50$ V)

OpenSPARC module	Logic gate count	Time, non-linear timing model	Time, linearized timing model	Speed-up	EDP_{95} sub-optimality: $EDP_{95,sub\text{-}opt}$
dec	4.0K	1.6 days	20 minutes	112X	0%
pmu	9.9K	3.4 days	40 minutes	121X	0%
pku	10.7K	4.1 days	50 minutes	119X	1.5%
gkt	10.6K	3.6 days	41 minutes	126X	0%
exu	19.5K	1 week	1.3 hours	124X	0%
lsu	46.5K	3 weeks	3.2 hours	160X	0%
tlu	69.5K	1 month	4.7 hours	162X	0.9%
fgu	104.1K	1.4 months	6.0 hours	172X	0.3%

TABLE IV
Processing Routes to Meet Delay Penalty Constraints and $PNMV \leq 0.001\%$ With $\Delta E \leq 5\%$ for All OpenSPARC Modules Simultaneously. In All Cases: $V_{\text{DD}} = 0.50$ V, $SNM_R = V_{\text{DD}}/6$, $p_{\text{Rm}} = 99.99\%$, Count-Limited Yield $\geq 99.999\%$

Delay penalty	5 nm node: IDC, p_m, p_{Rs}	7 nm node: IDC, p_m, p_{Rs}	10 nm node: IDC, p_m, p_{Rs}	14 nm node: IDC, p_m, p_{Rs}
≤ 5%	0.19, 0.8%, 2.2%	0.20, 0.9%, 2.3%	0.19, 0.9%, 2.6%	0.23, 0.9%, 2.6%
≤ 6%	0.25, 0.9%, 2.6%	0.26, 0.9%, 2.6%	0.26, 0.9%, 2.8%	0.31, 0.9%, 2.9%
≤ 7%	0.30, 0.9%, 2.7%	0.31, 0.9%, 2.8%	0.32, 0.9%, 2.9%	*0.35, 0.9%, 3.0%*
≤ 8%	*0.31, 0.9%, 2.7%*	*0.32, 0.9%, 2.8%*	*0.35, 0.9%, 2.9%*	*0.35, 0.9%, 3.0%*

Ideally, the same design point is selected using each of the two models (resulting in $EDP_{95,\text{sub-opt}} = 0\%$, example in Fig. 13). This is the case for five of the eight OpenSPARC modules (5 nm node), and the other three have $EDP_{95,\text{sub-opt}} \leq 2\%$ (Table III).[3] The linearized model achieves $>100\times$ speed-up in all cases.

B. CNT Processing and CNFET Circuit Design Guidelines

We now demonstrate the effectiveness of the gradient descent methodology to identify multiple sets of guidelines for processing parameters (i.e., processing routes) that meet design goals for all OpenSPARC modules simultaneously. For each OpenSPARC module, we first perform gradient descent (Fig. 11, with initial processing parameter values: $IDC = 0.50$, $p_m = 1\%$, $p_{\text{Rs}} = 4\%$, $p_{\text{Rm}} = 99.99\%$) to identify multiple acceptable design points (with delay penalty $\leq 5\%$, $PNMV \leq 0.001\%$, $\Delta E \leq 5\%$), and then we select the design point with the most relaxed processing requirements (though other selection criteria can be used, e.g., lowest EDP_{95}). Then, for each processing parameter, we select its most constrained value (i.e., the value closest to its ideal value: Table I) over all the selected design points (one for each OpenSPARC module). These values form a processing route, and we then validate that design goals are met for all modules for this processing route (e.g., using the nonlinear model to compute delay penalty). Table IV provides processing routes for the OpenSPARC modules at the 14, 10, 7, and 5 nm nodes (highlighted entries in Table IV are limited by the $PNMV$ constraint; other entries are limited by the delay penalty constraint). For each node, processing routes are shown for multiple delay penalty constraints to illustrate the tradeoff between delay penalty, $PNMV$, and processing requirements. All processing routes in Table IV meet count-limited yield $\geq 99.999\%$, resulting from minimum-width upsizing (step 2 in Fig. 3: to reach count-limited yield $\geq 99.9\%$) and CNT process

[3]In general, $EDP_{95,\text{sub-opt}}$ depends on the chosen set of design points since there is a finite number of possible values for $EDP_{95,\text{sub-opt}}$; results in Table III reflect a typical brute-force-based EDP optimization [52].

TABLE V
PROCESSING ROUTES TO MAINTAIN EDP BENEFIT ≥90% (VERSUS THE NOMINAL CASE) AND $PNMV \leq 0.001\%$ ($SNM_R = V_{DD}/6$) WITH $\Delta E \leq 5\%$ (ALL OPENSPARC MODULES). COUNT-LIMITED YIELD ≥99.999% IN ALL CASES

Node	V_{DD}	IDC	p_m	p_{Rs}	p_{Rm}
14 nm	0.50 V	0.35	0.9%	3.0%	99.99%
10 nm	0.50 V	0.33	0.9%	2.9%	99.99%
7 nm	0.50 V	0.31	0.9%	2.8%	99.99%
5 nm	0.50 V	0.30	0.9%	2.7%	99.99%
5 nm	0.45 V	0.29	0.9%	2.6%	99.99%
5 nm	0.40 V	0.27	0.9%	2.6%	99.99%
5 nm	0.35 V	0.25	0.9%	2.5%	99.99%

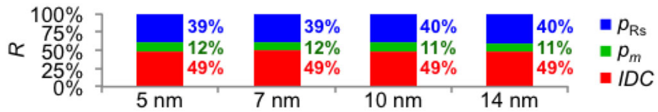

Fig. 14. Relative improvement of each processing parameter (versus $IDC = 0.50$, $p_m = 1\%$, $p_{Rs} = 4\%$) for processing routes in Table V ($V_{DD} = 0.50$ V). p_{Rm} is not shown since $p_{Rm} = 99.99\%$ in all cases in Table V.

improvements; if count-limited yield <99.999%, then we can return to step 2 in Fig. 3 to increase W_{MIN}, then repeat gradient descent (Fig. 11) to find processing routes.

We have so far targeted delay penalty ≤5%, $PNMV \leq 0.001\%$, and $\Delta E \leq 5\%$, which maintains ≥90% of the projected EDP benefits of nominal CNFET circuits despite CNT variations. However, achieving these design goals can impose processing requirements that may be difficult to achieve experimentally (e.g., $IDC = 0.19$ for delay penalty ≤5% at the 5 nm node: Table IV). In Table V, we provide alternative processing routes that maintain ≥90% of the projected EDP benefits of nominal CNFET circuits; we target design points with EDP benefit ≥90% (versus nominal) with a relaxed delay penalty constraint (≤10%, resulting in lower ΔE to meet the EDP benefit goal).

The amount by which each processing parameter is improved is a measure of its effectiveness to improve delay penalty and *PNMV* (gradient descent incurs larger updates for processing parameters that more significantly impact these performance metrics, details in [17, Sec. X-B]). Fig. 14 shows the relative improvement [R in (40)] of IDC, p_m, and p_{Rs} from their initial values to their final values (in Table V). R is calculated using the percentage improvement (I) and the total improvement (I_{Tot}) of the processing parameters

$$(I_{IDC}, I_{pm}, I_{pRs}) = \left(1 - \frac{IDC^{final}}{IDC^{init}},\ 1 - \frac{p_m^{final}}{p_m^{init}},\ 1 - \frac{p_{Rs}^{final}}{p_{Rs}^{init}}\right) \quad (38)$$

$$I_{Tot} = I_{IDC} + I_{pm} + I_{pRs} \quad (39)$$

$$(R_{IDC}, R_{pm}, R_{pRs}) = \left(\frac{I_{IDC}}{I_{Tot}},\ \frac{I_{pm}}{I_{Tot}},\ \frac{I_{pRs}}{I_{Tot}}\right). \quad (40)$$

The relative improvement is highest for *IDC* for all nodes, showing that *IDC* is a highly effective parameter to improve for reducing delay penalties and *PNMV* in an energy-efficient manner. From our results, we make the following conclusions.

1) The computationally efficient linearized timing model runs over 100× faster than the nonlinear timing model, and maintains sufficient accuracy to identify design points with $EDP_{95,\text{sub-opt}} \leq 2\%$ for all test cases.
2) $PNMV \leq 0.001\%$ can be efficiently computed.
3) Gradient descent is a systematic and scalable method to meet both delay penalty and *PNMV* constraints.
4) Gradient descent can efficiently identify multiple processing routes to meet design goals.
5) In contrast to traditional thinking (which focuses on reducing p_m to ultralow values), gradient descent identifies that reducing *IDC* is a highly effective means of meeting delay penalty and *PNMV* constraints, and that reducing p_m past 1% suffers from diminishing returns. Unlike trial-and-error approaches [52], gradient descent establishes these facts in a highly rigorous manner.

V. CONCLUSION

We have demonstrated a systematic methodology for joint exploration and optimization of CNT processing and CNFET circuit design to overcome the significant challenge of CNT variations. Our approach enables quick evaluation of delay variations and *PNMV* of CNFET VLSI circuits with >100× speed-up versus existing approaches. Our gradient descent-based framework accurately identifies the most important processing parameters, in conjunction with CNFET circuit sizing, to achieve high energy efficiency while satisfying circuit-level noise margin and yield constraints. Using this framework, an important question regarding CNT variations can be answered.

Question: What values of IDC, p_m, p_{Rs}, and p_{Rm} should be targeted for highly scaled VLSI CNFET circuits to maintain a significant portion of their projected speed and energy efficiency benefits despite CNT variations, while also meeting circuit-level noise margin and yield constraints?

Answer: At the 5 nm node, we recommend $IDC = 0.25$, $p_m = 0.9\%$, $p_{Rs} = 2.5\%$, and $p_{Rm} = 99.99\%$ to maintain ≥90% of the projected EDP benefits versus nominal CNFET circuits, with $PNMV \leq 0.001\%$, functional yield ≥99.999%, and $\Delta E \leq 5\%$. These processing guidelines are attractive since $p_m = 1\%$ and $p_{Rm} = 99.99\%$ have been experimentally demonstrated, $p_{Rs} = 4\%$ has been achieved, and promising work for continued improvement of p_{Rs} has been shown [19]. This leaves *IDC* to be improved by 2× (versus $IDC = 0.50$: shown experimentally), thus identifying CNT density variations as an important topic of research. Additionally, processing requirements may be further relaxed by combining various CNT processing techniques (e.g., CNT sorting [1] followed by VMR [32]). Processing routes for other nodes are provided in Table V.

Unlike existing trial-and-error techniques, our framework can systematically explore the large space of CNT processing options, and generate a variety of processing routes depending on CNT processing technology constraints. Such systematic exploration is essential for a successful CNFET technology to avoid potential obstacles. Future research directions include the following.

1) Incorporation of CNT-metal contact resistance variations and threshold voltage variations into our framework, as well as other CNT processing techniques (e.g., [19]).
2) Experimental validation of model parameters for high-density CNT growth techniques and for channel lengths closer to the ballistic regime [41], [43].
3) Examination of the applicability of our framework for other emerging nanotechnologies, as many emerging nanotechnologies are expected to exhibit substantial variations. Our methodology can be adapted to overcome challenges in those technologies as well.

ACKNOWLEDGMENT

The authors would like to thank STARnet SONIC, NSF, the Stanford Graduate Fellowship, and the Hertz Foundation

Fellowship (for M. M. Shulaker). They would also like to thank S. Keller, California Institute of Technology, S. Sinha, ARM, Inc., and Prof. D. Antoniadis, Massachusetts Institute of Technology, for valuable discussions.

REFERENCES

[1] M. S. Arnold, A. A. Green, J. F. Hulvat, S. I. Stupp, and M. C. Hersam, "Sorting carbon nanotubes by electronic structure using density differentiation," *Nat. Nanotech.*, vol. 1, no. 1, pp. 60–65, 2006.

[2] S. Bangsaruntin, G. M. Cohen, A. Majumdar, and J. W. Sleight, "Universality of short-channel effects in undoped-body silicon nanowire MOSFETs," *IEEE Electron Device Lett.*, vol. 31, no. 9, pp. 903–905, Sep. 2010.

[3] H. H. Bauschke and J. M. Borwein, "Dykstra's alternating projection algorithm for two sets," *J. Approx. Theory*, vol. 79, no. 3, pp. 418–443, 1994.

[4] G. Budiman, Y. Gao, X. Wang, S. Koswatta, and M. Lundstrom. (2014). *Cylindrical CNTMOSFET Simulator*. [Online]. Available: https://nanohub.org/resources/moscntr

[5] Q. Cao *et al.*, "Medium-scale carbon nanotube thin-film integrated circuits on flexible plastic substrates," *Nature*, vol. 454, pp. 495–500, Jul. 2008.

[6] L. Chang *et al.*, "IEDM short course," in *Proc. Int. Electron Device Meeting (IEDM)*, San Francisco, CA, USA, 2012.

[7] Z. Chen *et al.*, "An integrated logic circuit assembled on a single carbon nanotube," *Science*, vol. 311, no. 5768, p. 1735, 2006.

[8] C. Chu, "FLUTE: Fast lookup table based wirelength estimation technique," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2004, pp. 696–701.

[9] J. Deng *et al.*, "Carbon nanotube transistor circuits: Circuit-level performance benchmarking and design options for living with imperfections," in *Proc. Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, 2007, pp. 70–588.

[10] R. Dennard *et al.*, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE J. Solid-State Circuits*, vol. SC-9, no. 5, pp. 256–268, Oct. 1974.

[11] L. Ding *et al.*, "CMOS-based carbon nanotube pass-transistor logic integrated circuits," *Nat. Commun.*, vol. 3, Feb. 2012, Art. ID 677.

[12] (Jul. 2012). *Eigen C++ Template Library for Linear Algebra*. [Online]. Available: http://eigen.tuxfamily.org

[13] A. D. Franklin *et al.*, "Sub-10 nm carbon nanotube transistor," *Nano Lett.*, vol. 12, no. 2, pp. 758–762, 2012.

[14] A. Genz, "Numerical computation of multivariate normal probabilities," *J. Comput. Graph. Stat.*, vol. 1, no. 2, pp. 141–149, 1992.

[15] M. C. Hansen *et al.*, "Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering," *IEEE Des. Test*, vol. 16, no. 3, pp. 72–80, Jul. 1999.

[16] G. Hills *et al.*, "Rapid exploration of processing and design guidelines to overcome carbon nanotube variations," in *Proc. Design Autom. Conf.*, Austin, TX, USA, 2013, pp. 1–10.

[17] G. Hills *et al. Rapid Co-Optimization of Processing and Circuit Design to Overcome Carbon Nanotube Variations*. [Online]. Available: http://www.arxiv.org

[18] (2013). *ITRS*. [Online]. Available: http://www.itrs.net/Links/2013ITRS/Home2013.htm

[19] S. H. Jin *et al.*, "Using nanoscale thermocapillary flows to create arrays of purely semiconducting single-walled carbon nanotubes," *Nat. Nanotechnol.*, vol. 8, no. 5, pp. 347–355, 2013.

[20] S. J. Kang *et al.*, "High-performance electronics using dense, perfectly aligned arrays of single-walled carbon nanotubes," *Nat. Nanotechnol.*, vol. 2, no. 4, pp. 230–236, 2007.

[21] A. Khakifirooz *et al.*, "A simple semiempirical short-channel MOSFET current–voltage model continuous across all regions of operation and employing only physical parameters," *IEEE Trans. Electron Devices*, vol. 56, no. 8, pp. 1674–1680, Aug. 2009.

[22] J. Lohstroh *et al.*, "Worst-case static noise margin criteria for logic circuits and their mathematical equivalence" *IEEE J. Solid-State Circuits*, vol. 18, no. 6, pp. 803–807, Dec. 1983.

[23] J. Luo *et al.*, "A compact model for carbon nanotube field-effect transistors including non-idealities and calibrated with experimental data down to 9 nm gate length," *IEEE Trans. Electron Devices*, vol. 60, no. 6, pp. 1834–1843, Jun. 2013.

[24] D. Markovic, V. Stojanovic, B. Nikolic, M. A. Horowitz, and R. W. Brodersen, "Methods for true energy-performance optimization," *IEEE J. Solid-State Circuits*, vol. 39, no. 8, pp. 1282–1293, Aug. 2004.

[25] (Aug. 2014). *Nangate Open Cell Libraries*. [Online]. Available: http://www.nangate.com

[26] S. Nassif *et al.*, "High performance CMOS variability in the 65 nm regime and beyond," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Washington, DC, USA, 2007, pp. 569–571.

[27] (Dec. 2011). *OpenSPARC*. [Online]. Available: http://www.opensparc.net/opensparc-t2

[28] H. Park *et al.*, "High-density integration of carbon nanotubes via chemical self-assembly," *Nat. Nanotechnol.*, vol. 7, no. 12, pp. 787–791, 2012.

[29] J. Parker, C. Beasley, A. Lin, H.-Y. Chen, and H.-S. P. Wong, "Increasing the semiconducting fraction in ensembles of single-walled carbon nanotubes," *Carbon*, vol. 50, no. 14, pp. 5093–5098, 2012.

[30] N. Patil, J. Deng, A. Lin, H.-S. P. Wong, and S. Mitra, "Design methods for misaligned and mispositioned carbon-nanotube immune circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 10, pp. 1725–1736, Oct. 2008.

[31] N. Patil *et al.*, "Wafer-scale growth and transfer of aligned single-walled carbon nanotubes," *IEEE Trans. Nanotechnol.*, vol. 8, no. 4, pp. 498–504, Jul. 2009.

[32] N. Patil *et al.*, "VMR: VLSI-compatible metallic carbon nanotube removal for imperfection-immune cascaded multi-stage digital logic circuits using carbon nanotube FETs," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Baltimore, MD, USA, 2009, pp. 1–4.

[33] N. Patil *et al.*, "Scalable carbon nanotube computational and storage circuits immune to metallic and mispositioned carbon nanotubes," *IEEE Trans. Nanotechnol.*, vol. 10, no. 4, pp. 744–750, Jul. 2011.

[34] B. C. Paul *et al.*, "Impact of a process variation on nanowire and nanotube device performance," *IEEE Trans. Electron Devices*, vol. 54, no. 9, pp. 2369–2376, Sep. 2007.

[35] A. Raychowdhury *et al.*, "Variation tolerance in a multichannel carbon-nanotube transistor for high-speed digital circuits," *IEEE Trans. Electron Devices*, vol. 56, no. 3, pp. 383–392, Mar. 2009.

[36] J. A. Roy and I. L. Markov, "High-performance routing at the nanometer scale," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 6, pp. 1066–1077, Jun. 2008.

[37] R. Saito, G. Dresselhaus, and M. S. Dresselhaus, *Physical Properties of Carbon Nanotubes*. London, U.K.: Imperial College Press, 1998.

[38] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions*. Berlin, Germany: Springer, 1985.

[39] M. M. Shulaker *et al.*, "Carbon nanotube computer," *Nature*, vol. 501, no. 7468, pp. 526–530, 2013.

[40] M. M. Shulaker *et al.*, "Sensor-to-digital interface built entirely with carbon nanotube FETs," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 190–201, Jan. 2014.

[41] M. M. Shulaker *et al.*, "Carbon nanotube circuit integration up to sub-20 nm channel length," *ACS Nano*, vol. 8, no. 4, pp. 3434–3443, 2014.

[42] M. M. Shulaker *et al.*, "Monolithic 3D integration of logic and memory: Carbon nanotube FETs, resistive RAM, and silicon FETs," in *Proc. Int. Electron Devices Meeting (IEDM)*, San Francisco, CA, USA, 2014, pp. 27.4.1–27.4.4.

[43] M. M. Shulaker *et al.*, "High-performance carbon nanotube field-effect transistors," in *Proc. Int. Electron Devices Meeting (IEDM)*, San Francisco, CA, USA, 2014, pp. 33.6.1–33.6.4.

[44] S. Sinha *et al.*, "Design benchmarking to 7 nm with FinFET predictive technology models," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, Redondo Beach, CA, USA, 2012, pp. 15–20.

[45] (Apr. 2015). *Stanford University VSCNFET Model*. [Online]. Available: https://nano.stanford.edu/stanford-cnfet2-model

[46] L. Wei, D. J. Frank, L. Chang, and H.-S. P. Wong, "A non-iterative compact model for carbon nanotube FETs incorporating source exhaustion effects," in *Proc. Int. Electron Devices Meeting (IEDM)*, Baltimore, MD, USA, 2009, pp. 1–4.

[47] H. Wei *et al.*, "Monolithic three-dimensional integration of carbon nanotube FET complementary logic circuits," in *Proc. Int. Electron Devices Meeting (IEDM)*, Washington, DC, USA, 2013, pp. 19.7.1–19.7.4.

[48] N. H. Weste and D. M. Harris, *CMOS VLSI Design*. Boston, MA, USA: Pearson/Addison Wesley, 2005.

[49] J. Zhang, N. Patil, A. Hazeghi, and S. Mitra, "Carbon nanotube circuits in the presence of carbon nanotube density variations," in *Proc. 46th ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2009, pp. 71–76.

[50] J. Zhang, N. Patil, and S. Mitra, "Probabilistic analysis and design of metallic-carbon-nanotube-tolerant digital logic circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 9, pp. 1307–1320, Sep. 2009.

[51] J. Zhang *et al.*, "Carbon nanotube correlation: Promising opportunity for CNFET circuit yield enhancement," in *Proc. 47th Design Autom. Conf. (DAC)*, Anaheim, CA, USA, 2010, pp. 889–892.

[52] J. Zhang, N. Patil, H.-S. P. Wong, and S. Mitra, "Overcoming carbon nanotube variations through co-optimized technology and circuit design," in *Proc. Int. Electron Devices Meeting (IEDM)*, Washington, DC, USA, 2011, pp. 4.6.1–4.6.4.

[53] J. Zhang, "Variation-aware design of carbon nanotube digital VLSI circuits," Ph.D. dissertation, Dept. Electr. Eng., Stanford Univ., Stanford, CA, USA, 2011.

[54] J. Zhang *et al.*, "Carbon nanotube robust digital VLSI," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 4, pp. 453–471, Apr. 2012.

Gage Hills received the B.S. degree in electrical engineering and computer science from Yale University, New Haven, CT, USA, in 2007, and the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2012, where he is currently pursuing the Ph.D. degree.

His current research interests include experimental demonstrations and statistical modeling of very large-scale carbon nanotube-based digital circuits.

Jie Zhang (S'05–M'11) received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2006, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 2008 and 2011, respectively.

Since 2011, he has been with Google, Inc., Mountain View, CA, USA. His current research interests include modeling and simulation of carbon nanotube-based devices and circuits, with a focus on variation-aware design and optimization.

Dr. Zhang was a recipient of the Stanford Graduate Fellowship and the Top-Class Scholarship from Tsinghua University.

Max Marcel Shulaker received the B.S. in electrical engineering from Stanford University, Stanford, CA, USA, in 2011, where he is currently pursuing the Ph.D. degree, and researches on experimentally demonstrating nanosystems with emerging technologies.

His current research interests include realizing increased levels of integration for carbon nanotube-based digital logic circuits.

Mr. Shulaker is a Stanford Graduate Fellow and a Fannie and John Hertz Fellow.

Hai Wei (S'09) received the B.S. (Hons.) degree in microelectronics from Tsinghua University, Beijing, China, in 2007, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 2010 and 2014, respectively.

His current research interests include design and fabrication of carbon nanotube field-effect transistors and circuits and monolithic 3-D integrated circuits.

Mr. Wei was a recipient of the Stanford School of Engineering Fellowship Award.

Chi-Shuen Lee received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2011, and the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2014, where he is currently pursuing the Ph.D. degree.

He is currently involved in the compact modeling of carbon nanotube field-effect transistors and performance benchmarking of digital systems based on emerging CMOS technologies in the sub-10 nm regime. His current research interests include modeling and simulation of nanoscale transistors and performance benchmarking.

Arjun Balasingam received the Diploma degree from Archbishop Mitty High School, San Jose, CA, USA, in 2014, as the Valedictorian of his class. He is currently pursuing the B.S. degree with the School of Engineering, Stanford University, Stanford, CA, USA.

His current research interests include electrical engineering, mathematics, and computer science.

Mr. Balasingam was selected as a Siemens Competition Regional Finalist in 2012 and a Semifinalist in 2013 for his work on a variety of engineering research projects. He was selected as an Intel Science Talent Search Semifinalist in 2014. He was a recipient of the IEEE Silicon Valley Engineering Council Education Award in 2014.

H.-S. Philip Wong (F'01) received the B.Sc. (Hons.) degree from the University of Hong Kong, Hong Kong, the M.S. degree from Stony Brook University, New York, NY, USA, and the Ph.D. degree from Lehigh University, Bethlehem, PA, USA.

From 1988 to 2004, he was with IBM T.J. Watson Research Center, Yorktown Heights, NY, USA, where he has been Research Staff Member, Manager, and Senior Manager. As Senior Manager, he was responsible for shaping and executing IBM's strategy on nanoscale science and technology as well as exploratory silicon devices and semiconductor technology. He joined Stanford University, Stanford, CA, USA, as a Professor of Electrical Engineering, in 2004, where he is the Willard R. and Inez Kerr Bell Professor with the School of Engineering. His academic appointments include the Chair of Excellence of the French Nanosciences Foundation, Grenoble, France, a Guest Professor with Peking University, Beijing, China, an Honorary Professor with the Institute of Microelectronics of Chinese Academy of Sciences, Beijing, a Visiting Chair Professor of Nanoelectronics with Hong Kong Polytechnic University, Hong Kong, and the Honorary Doctorate degree from the Institut Polytechnique de Grenoble, Grenoble. His current research interests include carbon electronics, 2-D layered materials, wireless implantable biosensors, directed self-assembly, nanoelectro mechanical relays, device modeling, brain-inspired computing, and nonvolatile memory devices such as phase change memory and metal oxide resistance change memory.

Prof. Wong has served as an Elected Member of the Electron Devices Society AdCom from 2001 to 2006, an Editor-in-Chief of the IEEE TRANSACTIONS ON NANOTECHNOLOGY from 2005 to 2006, the Sub-Committee Chair of the International Solid-State Circuit Conference from 2003 to 2004, the General Chair of the International Electron Devices Meeting in 2007. He is the IEEE ExCom Chair of the Symposia of Very Large-Scale Integration Technology and Circuits.

Subhasish Mitra (F'13) directs the Robust Systems Group in the Department of Electrical Engineering and the Department of Computer Science of Stanford University, where he is the Chambers Faculty Scholar of Engineering. Before joining Stanford, he was a Principal Engineer at Intel.

Prof. Mitra's research interests include robust systems, VLSI design, CAD, validation and test, emerging nanotechnologies, and emerging neuroscience applications. His X-Compact technique for test compression has been key to cost-effective manufacturing and high-quality testing of a vast majority of electronic systems, including numerous Intel products. X-Compact and its derivatives have been implemented in widely-used commercial Electronic Design Automation tools. His work on carbon nanotube imperfection-immune digital VLSI, jointly with his students and collaborators, resulted in the demonstration of the first carbon nanotube computer, and it was featured on the cover of NATURE. The NSF presented this work as a Research Highlight to the US Congress, and it also was highlighted as "an important, scientific breakthrough" by the BBC, Economist, EE Times, IEEE Spectrum, MIT Technology Review, National Public Radio, New York Times, Scientific American, Time, Wall Street Journal, Washington Post, and numerous other organizations worldwide.

Prof. Mitra's honors include the Presidential Early Career Award for Scientists and Engineers from the White House, the highest US honor for early-career outstanding scientists and engineers, ACM SIGDA/IEEE CEDA A. Richard Newton Technical Impact Award in Electronic Design Automation, "a test of time honor" for an outstanding technical contribution, and the Intel Achievement Award, Intel's highest corporate honor. He and his students published several award-winning papers at major venues: IEEE/ACM Design Automation Conference, IEEE International Solid-State Circuits Conference, IEEE International Test Conference, IEEE Transactions on CAD, IEEE VLSI Test Symposium, Intel Design and Test Technology Conference, and the Symposium on VLSI Technology. At Stanford, he has been honored several times by graduating seniors "for being important to them during their time at Stanford."

Prof. Mitra has served on numerous conference committees and journal editorial boards. He served on DARPA's Information Science and Technology Board as an invited member. He is a Fellow of the ACM and the IEEE.