Received: 17 December 2019 | Revised: 30 August 2020 | Accepted: 3 November 2020

DOI: 10.4218/etrij.2019-0556

ORIGINAL ARTICLE

ETRI Journal WILEY

An efficient reliability estimation method for CNTFET-based logic circuits

Hadi Jahanirad | **Mostafa Hosseini**

Department of Electrical Engineering, University of Kurdistan, Sanandaj, Iran

Correspondence
Hadi Jahanirad, Department of Electrical Engineering, University of Kurdistan, Sanandaj, Iran.
Email: h.jahanirad@uok.ac.ir

Carbon nanotube field-effect transistors (CNTFETs) have been widely studied as a promising technology to be included in post-complementary metal-oxide-semiconductor integrated circuits. Despite significant advantages in terms of delay and power dissipation, the fabrication process for CNTFETs is plagued by fault occurrences. Therefore, developing a fast and accurate method for estimating the reliability of CNTFET-based digital circuits was the main goal of this study. In the proposed method, effects related to faults that occur in a gate's transistors are first represented as a probability transfer matrix. Next, the target circuit's graph is traversed in topological order and the reliabilities of the circuit's gates are computed. The accuracy of this method (less than 3% reliability estimation error) was verified through various simulations on the ISCAS 85 benchmark circuits. The proposed method outperforms previous methods in terms of both accuracy and computational complexity.

KEYWORDS

CNTFETs, gate-level circuit design, reliability estimation algorithms, transistor-level design

1 | INTRODUCTION

The carbon nanotube field-effect transistor (CNTFET) is one of the most promising transistor types that can replace metal-oxide-silicon field-effect transistors (MOSFETs) in future digital integrated circuits (ICs) [1,2]. Since the introduction of complementary metal-oxide-semiconductor (CMOS) technology, the downscaling of MOSFETs has been pursued continuously to keep pace with Moore's law [3,4]. In sub-micron CMOS technology (<100 nm transistor channel length), quantum mechanical effects such as electron tunneling through channels and thin insulator films have become sources of some undesirable phenomena [5–7]. Increased power dissipation, defect rates, and significant process variations represent significant barriers to achieving enhanced performance using sub-32 nm technologies. CNTFETs, as promising alternatives to MOSFETs, have some excellent advantages (eg, near-ballistic transport properties, high carrier mobility (10^3 to 10^4 $cm^2/V\cdot s$), and easy integration of high-k dielectric materials) [8–10]. Based on these properties, CNTFET-based ICs exhibit significant advantages in terms of power consumption and delay [11–15].

CNTFET-based implementations of various digital modules, such as inverters and NAND, NOR, and SRAM cells, have been reported in previous studies [16–19].

One important challenge in CNTFET-based ICs is reliability. With progressive downscaling, the control of device features (eg, CNT diameter or the alignment of CNTs) becomes increasingly complex. This results increased transistor defect rates, as well as the production of faulty gates and IC reliability degradation. Several studies have focused on probabilistically modeling the defects generated in CNTFETs (eg, open and short defects) [20–24]. Based on these models, the failure probabilities of various gates can be extracted. If the error probabilities of a circuit's gates are determined, then the circuit's reliability can be analyzed using conventional gate-level estimators. Multi-valued logic must be handled properly to apply conventional reliability evaluators to CNTFET-based ICs [25,26]. For example, when the pull-up and pull-down networks in a logic gate turn off (or on) simultaneously based on fault occurrence, the output state will be neither “0” nor “1.” Instead, a “FLOAT” (or “Tri-STATE”) state will be generated. In this paper, we propose a reliability evaluator based on a probability transfer matrix (PTM). In this method, four states (“0,” “1,” “Tri-STATE,” and “FLOAT”) are defined for each node in a circuit and the transition probabilities among these states are computed and propagated by traversing the circuit's graph.

The reliability analysis of CNTFET-based logic circuits has been investigated in previous studies. In [26], a pseudo-complementary CNTFET-based multi-valued logic model was proposed and a stochastic approach was applied for reliability computation. In this method, each gate is replaced with a multiplexer in which the selectors connect to the gate's inputs and the multiplexer's inputs contain random sequences including three symbols of “0,” “2,” and “1,” which represent the logic states of “0,” “1,” and “Tri-STATE,” respectively. The execution time of this method is very high because of its simulation-based nature. Another probabilistic approach can be found in [27], where gate output failures were divided into three cases: a gate's inputs are faulty but the gate is fault-free, a gate is faulty but gate's inputs are fault-free, and both a gate and its inputs are faulty simultaneously. This method is very fast, but it encounters two major problems. First, CNTFET fault modeling is inaccurate because it is assumed that the CNTFET always turns on when the “Tri-STATE” signal is applied to its gate terminal. In a more realistic model, the transistor may turn on or turn off according to its chirality. The second problem is related to reconvergent fan-out signals, which are a source of inaccuracy in this method (additional descriptions can be found in Section 4). Sirinivasu and Sridharan [28] developed a PTM-based approach for CNTFET-based circuit reliability analysis. Based on open and short faults, an error probability matrix is derived for each circuit gate. In this matrix, for every input vector (matrix row), the probabilities of generating of “0,” “1,” and “Tri-STATE” states in a gate's output are computed. The matrix is then reduced to a conventional matrix without “Tri-STATE” entries. Finally, the conventional PTM method is applied. However, this approach cannot handle reconvergent fan-outs properly and the removal of “Tri-STATE” entries in the PTM matrix results in undesirable effects.

The main contributions of this paper can be summarized as follows. First, we develop a reliability evaluator for combinational logic circuits. Second, we derive gate failure probabilities using transistor-level topology. Third, we compare the reliabilities of various design methods for CNTFET-based primitive gates.

This remainder of this paper is organized as follows. CNTFET-based gate failure modeling is reviewed in Section 2. In Section 3, the proposed method for the reliability evaluation of logic gates is described, followed by a description of the reliability evaluation flow for combinational logic circuits in Section 4. Simulation results are presented in Section 5 and our conclusions are summarized in Section 6.

2 | CNTFET-BASED GATE FAILURE MODELING

Since the invention of CNTFETs in 1998, based on the fault-prone nature of these devices, many researchers have studied their reliability [29]. Depending on its chirality, a CNT can be a metal or semiconductor. The former type of CNT is the main reason for fault generation in CNTFETs [23]. Generally, there are two main fault types in CNTFETs. The first is a short fault, which occurs when a metallic CNT grows between the source and drain terminals. The second is an open fault, which occurs when no CNTs remain between the source and drain terminals following the chemical removal of metallic CNTs [20]. Figure 1 presents three possible cases that can occur during the CNTFET synthesis process. The CNTFETs in Figure 1A and 1B are functional, while those in Figure 1C and 1D contain short and open failures, respectively. Variations in CNT diameter and density result in delays and power consumption variation. Additionally, the misalignment of CNTs can lead to incorrect functionality [30].

Several statistical failure analyses have been performed on CNTFETs [20–24]. In these studies, the distributions of CNT counts and CNT spacing in a CNTFET were defined (typically as geometric probability density functions). Next, based on these distributions, the probabilities of fault occurrences (open and short faults) were derived for CNTFETs containing N CNTs [31,32]. A similar approach was adopted in [20], but a binomial distribution was applied to the CNT counts.

One major barrier to incorporating CNTFETs in very-large-scale integration (VLSI) chips is a high metallic CNT growth rate (30%) during the synthesis process [22]. Various metallic CNT removal methods can be applied to reduce this ratio, but these methods may inadvertently remove some semiconductor CNTs [32,33]. To compute the short and open failure probabilities (P_S and P_O,

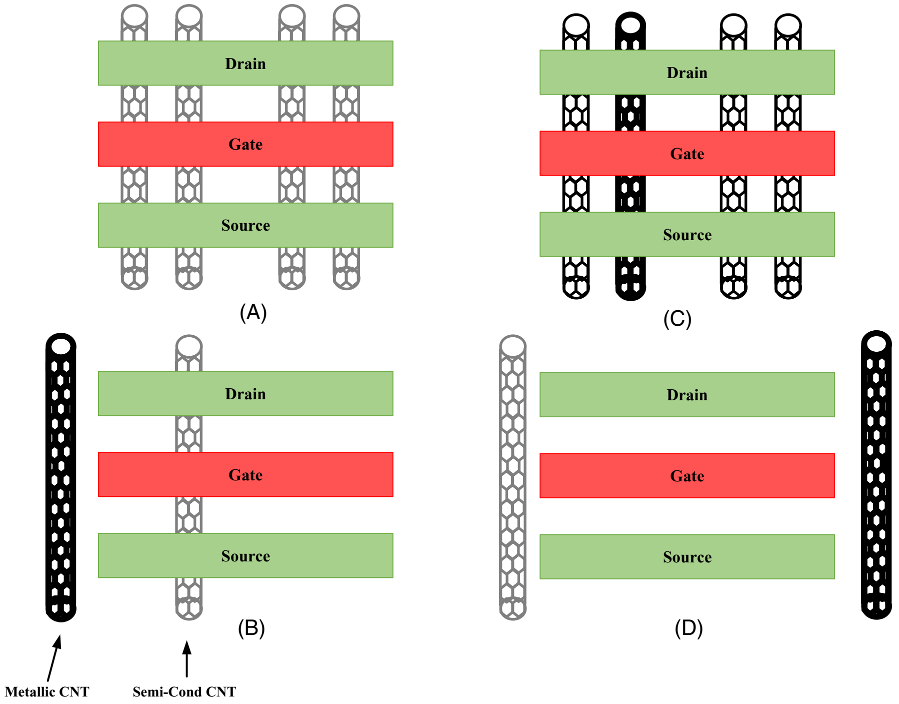


FIGURE 1 Various structures of CNTFETs. (A) Functional CNTFET containing four CNTs, (B) functional CNTFET containing one CNT, (C) CNTFET containing metallic CNTs (short fault), (D) CNTFET containing no CNTs (open fault)

respectively) of a CNTFET, we adopt the analytical model that was developed in [26]. In this model, for a CNTFET containing N_{CNT} CNTs, P_S and P_O can be computed according to (1) and (4), respectively.

$$P_S(k) = \sum_{k_1=1}^{k} \binom{k}{k_1} (1 - p_{mr}^{k_1}) p_m^{k_1} (1 - p_m)^{k-k_1} . \tag{1}$$

$$\overline{P_S} = \sum_{k=1}^{2\overline{N_{CNT}}} \binom{2\overline{N_{CNT}}}{k} (\frac{1}{2})^{2\overline{N_{CNT}}} P_S(k) . \tag{2}$$

$$P_O(k) = \sum_{k_1=1}^{k} \binom{k}{k_1} (p_{sr}^{k-k_1} p_{mr}^{k_1}) p_m^{k_1} (1 - p_m)^{k-k_1} . \tag{3}$$

$$\overline{P_O} = \sum_{k=1}^{2\overline{N_{CNT}}} \binom{2\overline{N_{CNT}}}{k} (\frac{1}{2})^{2\overline{N_{CNT}}} P_O(k) . \tag{4}$$

In these equations, P_m is the probability of a CNT being metallic and the probabilities of metallic and semiconductor CNTs being removed during the removal process are denoted as P_{mr} and P_{sr}, respectively. Ultimately, the average probability of fault occurence in the CNTFET ($\overline{P_F}$) would be calculated according to (5).

$$\overline{P_F} = \overline{P_S} + \overline{P_O}. \tag{5}$$

One major problem encountered by CNTFET-based VLSI chips is the high open and short fault probabilities of CNTFETs. One promising technique was proposed in [24] to improve the reliability of CNTFETs to an acceptable level. In a so-called asymmetrically correlated CNT (ACCNT), a row containing c transistors with series wiring (Figure 2) is fabricated. In this configuration, a short fault is generated when all transistors in the row are shorted, meaning the short fault probability of a row can be calculated according to (6).

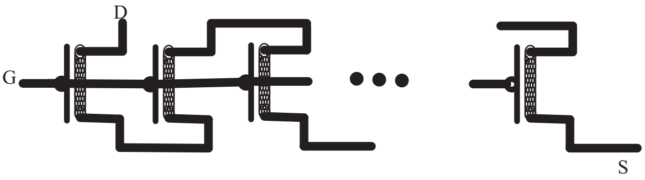


FIGURE 2 ACCNT row circuitry

$$\overline{P_{S\text{ - }ACCNT}} = (\overline{P_{S\text{ - }CNTFET}})^c . \tag{6}$$

In contrast, an open fault only requires one open CNTFET. This statement is the complement of the statement "all CNTFETs are not open." Therefore, the average open fault probability of a row in an ACCNT can be computed according to (7).

$$\overline{P_{O\text{ - }ACCNT}} = 1 - (1 - \overline{P_{O\text{ - }CNTFET}})^c . \tag{7}$$

Simulation results reveal a dramatic reduction in the gate short fault probability for ACCNTs, but some aspects of ACCNTs, such as the area overhead and additional power consumption, reduce efficiency [24].

3 | GATE RELIABILITY EVALUATION

Two main factors should be determined during the reliability evaluation of a logic gate. The first one is how open and short faults in a gate's CNTFETs can generate faulty values in that gate's outputs. The second is how faulty values generated by other gates can propagate to the outputs of the target gate.

In this section, we present an effective approach to analyzing these factors. First, the effects of CNTFET-related faults are represented using a PTM. Next, the transformation probabilities from one state to the other states in the gate's inputs are represented by a signal transition probability matrix (STPM). Based on a joint probability input matrix (JPIM) and the gate's PTM, the STMP of the gate's output is computed. The reliability of the gate is calculated by summing the probabilities of transitions from state "0" to state "0" and from state "1" to state "1" in the STPM.

3.1 | Gate PTM computation

For an arbitrary gate designed based on complementary, ratioed, and dynamic methodologies, there is at least one path toward V_{dd} (the pull-up network (PUN)) and at least one path toward GND (the pull-down network (PDN)). In some design methodologies such as pass-transistors or transmission gate design styles, the PUN and PDN are determined by the input states. In a fault-free gate, when the "1" or "0" states are applied to the inputs, only one of the PUN or PDN turns on while the other turns off, meaning the gate's outputs can only take on binary values ("1" or "0"). In contrast, based on probable open or short fault occurrence, the PUN and PDN may turn on ("Tri-STATE") or off ("FLOAT"), simultaneously.

To generate the "0" state at the gate's output, at least one path between the output and GND should turn on and all paths from V_{dd} to the output should turn off. To generate the "1" state, the inverse conditions must be satisfied. Regarding the "Tri-STATE" state, there should be at least one turned on path toward V_{dd} and at least one turned on path toward GND. For the "FLOAT" state, all paths to V_{dd} and GND should be turned off simultaneously.

To compute the PTM, we should determine the probabilities of the "0," "1," "Tri-STATE," and "FLOAT" states for each input vector. For a gate with N_{inp} inputs, the size of the PTM matrix is $3^{\text{Ninp}} \times 3$. For each input vector (an input can take on "0," "1," and "Tri-STATE" states), the output is a "0," "1," or "Tri-STATE" state with varying probabilities. The "FLOAT" state is included in the PTM as follows. When a node transforms from a state A into the "FLOAT" state, the logic of that node remains in state A [25]. Consequently, for a gate's PTM construction, the transition probability to the "FLOAT" state is divided by three and added to the probabilities of three other states ("0," "1," and "Tri-STATE").

Before continuing with our analysis, we define the conditions under which a CNTFET can act as a closed switch (turning on) or open switch (turning off). In the following discussion, we only consider an n-type CNTFET (N-CNTFET) because a p-type CNTFET (P-CNTFET) can be treated as a dual-n-type transistor. Suppose that the gate terminal of an N-CNTFET is connected to the logic "1." This transistor turns on when it is in normal mode or suffers from a short fault (N/S). In contrast, an open fault (O) would turn this transistor off. If the gate terminal state becomes "0," then an n-type CNTFET only turns on when the transistor contains a short fault (S) and turns off when the transistor is fault-free or contains an open fault (N/O).

The case of applying the "Tri-STATE" logic to a CNTFET's gate terminal requires additional effort. First, it must be determined whether or not the "Tri-STATE" voltage level (V_{Tri}) can turn on the transistor. This depends on the V_{th} value of the transistor and the value of V_{Tri}. If $V_{Tri} < V_{th}$, then the "Tri-STATE" turns off the N-CNTFET. Otherwise, the transistor turns on the N-CNTFET. As indicated in (8), various factors determine V_{th}, where a = 2.49 Å is the distance between carbon atoms, $V_\pi = 3.033$ eV is the carbon π-π bond energy, e is the electron charge, and d_{CNT} is the CNT diameter

$$V_{th} = \frac{\sqrt{3}aV_\pi}{3ed_{CNT}}. \tag{8}$$

$$d_{CNT} = \frac{\sqrt{3}a_0}{\pi}\sqrt{n^2 + m^2 + nm}. \tag{9}$$

Among these parameters, only d_{CNT} is variable and is defined by the CNT chirality vector (**m**, **n**) according to (9). For example, two CNTFETs with chirality vectors (19, 0) and (13, 0) have threshold voltages equal to 0.293 and 0.428 V, respectively.

A logic gate is designed such that the drive strengths of the PUN and PDN are equal. Therefore, if all the paths in the PUN and PDN turn on simultaneously, then the output voltage should be V_{dd} / 2. If there are some paths that turn off in the PUN and PDN, then the value of V_{Tri} will deviate from V_{dd} / 2. For example, suppose that only one path in the PUN is turned off while all paths in the PDN are turned on. This situation leads to greater drive strength in the PDN compared to the PUN, so the generated voltage in the gate's output would be $V_{dd} / 2 - \Delta$V. In this expression, ΔV is a positive voltage that is related to the difference between the PUN and PDN drive strengths. When additional paths are turned off in the PUN, the value of ΔV increases, resulting in a lower V_{Tri} value. The precise value of V_{Tri} should be determined via

HSPICE simulations of various situations that can occur in the PUN and PDN.

In our method, we simply consider a single "Tri-STATE" state in the PTM that represents all generated voltage levels. This is a source of error in our method because when we apply the "Tri-STATE" to the input of a gate, we should know which voltage level to consider for V_{Tri}.

Handling this problem in the proposed method is accomplished as follows. Considering the various scenarios that can result in the generation of a "Tri-STATE" in a gate's output, it is possible that V_{Tri} will take on a value in the range of 0 V to V_{dd}. We assume that this value follows a uniform distribution. Consequently, when the "Tri-STATE" is applied to the input of a gate, meaning the gate terminals of the related CNTFETs are connected to V_{Tri}, we should consider all possible cases. Suppose that the gate terminals of four CNTFETs (two P-CNTFETs from the PUN and two N-CNTFETs from the PDN) are connected to V_{Tri}. Then, we divide the [0, V_{dd}] voltage range according to the threshold voltages of these four CNTFETs. An example is presented in Figure 3, where five regions (R_1, R_2, …, R_5) are defined. In R_1, R_2, and R_3, the N-CNTFETs are turned on ($V_{\mathrm{Tri}} > V_{\mathrm{th,I}}$ (N-CNTFET)). In R_3, R_4, and R_5, the P-CNTFETs are turned on ($V_{\mathrm{Tri}} > |V_{\mathrm{th,j}}$ (P-CNTFET)|). The statuses of P-CNTFET(1) and P-CNTFET(2), as well as N-CNTFET(1) and N-CNTFET(2), are represented in Figure 3. Once the statuses (turning on or off) of the CNTFETs are determined for each region, we then handle a turned on (or turned off) N-CNTFET similar to the situation where V_{dd} (or GND) is connected to its gate terminal. It should be noted that the probability that is calculated for a region R_{j} must be scaled by the probability of V_{Tri} falling in this region. This probability is calculated according to (10), where $V_{\max}(j)$ and $V_{\min}(j)$ are the lower and upper voltage levels defining the region R_{j}.

$$P_{\mathrm{scale}}(R_j) = \frac{(V_{\max}(j) - V_{\min}(j))}{V_{\mathrm{dd}}}. \tag{10}$$

It is also worth noting that all transistors (similar types of CNTFETs) in the same column in the gate layout have similar V_{th} values because similar CNTs contribute to their structures [24].

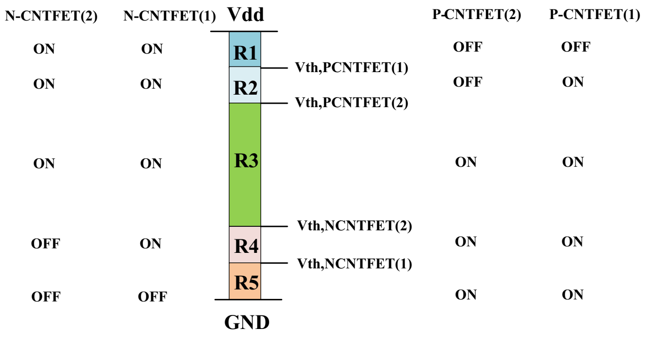


FIGURE 3 Region definitions for "Tri-STATE" handling

To activate (turn on) a path from the output to GND or from the output to V_{dd}, all transistors in the path should turn on. Unlike in the pass-transistor and transmission gate designs, the input voltage is applied to the CNTFET's gate terminal. In this case, the probability of turning on path i can be calculated according to

$$P_{\text{path - on}}(i) = \prod_{\mathrm{tr}=1}^{N_{\mathrm{path}}} P_{\text{tr - on}}(\mathrm{tr}). \tag{11}$$

Based on the state of the input connected to the CNTFET's gate terminal, $P_{\text{tr-on}}(\mathrm{tr})$ may be P_{S} (in case S) or $1 - P_{\mathrm{o}}$ (in case N/S). To deactivate (turn off) a path, at least one of the path's transistors must be turned off. Therefore, we use (12) to calculate the path deactivation probability.

$$P_{\text{path - off}}(i) = 1 - P_{\text{path - on}}(i). \tag{12}$$

If there is more than one path between the supply rail (V_{dd} or GND) and the gate's output (eg, N_{path}), the activation probabilities of all paths are computed. Then, the disconnection and connection probabilities of the two nodes (supply rail and output nodes) are calculated using (13) and (14), respectively. In these equations, up/down represents all paths from the gate's output to V_{dd}/GND.

$$P_{\mathrm{conn}}(\mathrm{up/down}) = 1 - P_{\mathrm{disc}}. \tag{13}$$

$$P_{\mathrm{disc}}(\mathrm{up/down}) = \prod_{i=1}^{N_{\mathrm{path}}} P_{\text{path - off}}(i). \tag{14}$$

There can be some paths in the PUN or PDN that share a CNTFET. An example is presented in Figure 4, where the T1 CNTFET is shared between paths 1 and 2. In this case, we combine these two paths into one super path, where the super path's activation probability is calculated according to (15). According to this equation, the super path turns on when the shared transistor is ON and at least one of the unshared transistors in ON (this case is equivalent to the complement of all unshared transistor being OFF simultaneously).

$$P_{\text{super - path}}(\mathrm{on}) = P_{\text{tr - on}}(1) \times (1 - (1 - P_{\text{tr - on}}(2)) \times (1 - P_{\text{tr - on}}(3))). \tag{15}$$

Generally, if M paths share N_{shared} CNTFETs and the kth path includes $N_{\mathrm{p}}(k)$ unshared CNTFETs, then the probability of turning on the super path is computed according to (16).

$$P_{\text{super - path}}(\mathrm{on}) = \prod_{i=1}^{N_{\mathrm{Shared}}} P_{\text{tr - on}}(i) \times (1 - \prod_{k=1}^{M} (1 - \prod_{j=1}^{N_{\mathrm{p}}(k)} P_{\text{tr - on}}(j))). \tag{16}$$

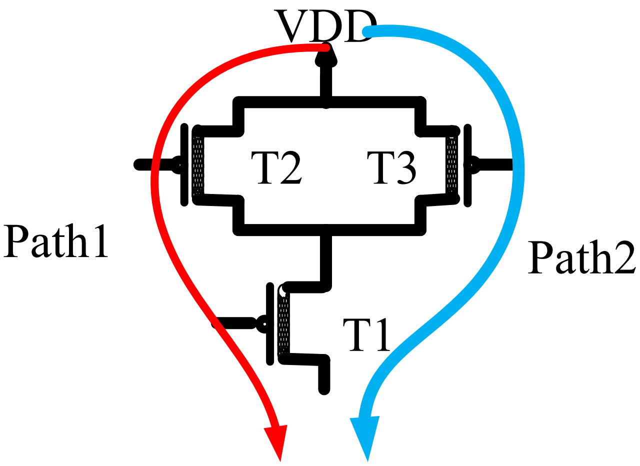


FIGURE 4 A super path topology

Regarding to the gate's PTM computation, for row r (application of the rth input vector), P_{conn} and P_{disc} should be calculated before the following steps are implemented. For the generation of the "FLOAT" state at the gate's output, the up and down paths must be disconnected according to (17).

$$P_{\text{FLOAT}}(r) = P_{\text{disc - up}}(r) \times P_{\text{disc - down}}(r). \qquad (17)$$

For the case of the "0" ("1") state, all up (down) paths should be disconnected and at least one down (up) path should be connected. Furthermore, the sharing of $P_{\text{FLOAT}}(r)$ should be considered according to (18) (or (19)).

$$P_0(r) = P_{\text{disc - up}}(r) \times P_{\text{conn - down}}(r) + \frac{1}{3} \times P_{\text{FLOAT}}(r). \qquad (18)$$

$$\mathrm{P}_1(r) = \mathrm{P}_{\text{conn - up}}(r) \times \mathrm{P}_{\text{disc - down}}(r) + \frac{1}{3} \times \mathrm{P}_{\text{FLOAT}}(r). \qquad (19)$$

Finally, for the "Tri-STATE" case, at least one up path and one down path should be turned on and the sharing of the "FLOAT" state also should be included according to (20).

$$P_{\text{tri - state}}(r) = P_{\text{conn - up}}(r) \times P_{\text{conn - down}}(r) + \frac{1}{3} \times P_{\text{FLOAT}}(r). \qquad (20)$$

After computing (17) to (20) for all rows ($r = 0$ to $r = 3^{\text{Ninp}} - 1$), the gate's PTM computation is completed.

3.2 | STPM

In a fault-free circuit, the output of a gate can be in the "0" or "1" states. However, when open and short faults are introduced, the circuit's nodes can take on four states ("0," "1," "Tri-STATE," and "FLOAT"). Generation of the "Tri-STATE" and "FLOAT" states in a gate's outputs can be caused by the propagation of such states that are previously generated in the gate's inputs or by the occurrence of an open/short fault in the gate's CNTFETs. However, for a fault-free gate, the "Tri-STATE" or "FLOAT" states generated in the gate's inputs can still be propagated to its outputs. For example, consider a NOT gate, where the PUN and PDN consist of a single P-CNTFET and a single N-CNTFET, respectively. Additionally, suppose the related voltage of a "Tri-STATE" state in the input can turn on both the P-CNTFET and N-CNTFET transistors. In this case, the "Tri-STATE" state will turn on both the PDN and PUN, so the generated state in the gate's output will be the "Tri-STATE" state. If the "Tri-STATE" state cannot turn on both the P-CNTFET and N-CNTFET, then a "FLOAT" state will be generated in the gate's output.

The probability of occurrence of the "FLOAT" state is merged into the other states, as discussed in Section 3.1. For each node in the circuit, we define an STPM according to (21). In this context, a transition in a node indicates transforming from one state A to another state B based on a possible fault occurrence.

$$\text{STPM} = \begin{bmatrix} p(s_0 \to s_0) & p(s_0 \to s_{\text{TS}}) & p(s_0 \to s_1) \\ p(s_{\text{TS}} \to s_0) & p(s_{\text{TS}} \to s_{\text{TS}}) & p(s_{\text{TS}} \to s_1) \\ p(s_1 \to s_0) & p(s_1 \to s_{\text{TS}}) & p(s_1 \to s_1) \end{bmatrix}. \qquad (21)$$

The rows/columns of this matrix represent a node's signal states before/after a transition. The first, second, and third rows/columns are dedicated to the "0," "Tri-STATE," and "1" states of the node, respectively. For example, STPM(1, 2) represents the probability of a transition from the correct "0" state to the incorrect "Tri-STATE" state. The STPM contains the reliability information for a gate, which is calculated as the sum of the probabilities of correct "0" (STPM(1, 1)) states and correct "1" (STPM(3, 3)) states, as shown in (22).

$$\text{Rel}_{\text{gate}} = \text{STPM}(1, 1) + \text{STPM}(3, 3). \qquad (22)$$

3.3 | Gate STPM computation

We have adopted the basic PTM-based method discussed in [34], which was developed for binary logic values, for our four-valued logic setting. Suppose that a gate has N_{inp} inputs and one output. The first step in STPM computation is constructing a JPIM. The JPIM is a $3^{\text{Ninp}} \times 3$ matrix that is constructed by combining the STPMs of all inputs in N_{inp} steps. Each entry in this matrix represents the occurrence probability of an input vector. In the first stage, STPM_1 (index represents the index of the related input) is combined with STPM_2 according to (23), where (u_1, v_1) indices are related to STPM_1 and (u_2, v_2) indices are related to STPM_2. These variables can take on states of "0," "Tri-STATE," or "1."

$$\text{JPIM}_{1,2}(u_1, v_1, u_2, v_2) = p(s_{u1} \rightarrow s_{v1}) \times p(s_{u2} \rightarrow s_{v2})\,. \quad (23)$$

In this operation, each entry in STPM_1 is multiplied by the STPM_2 matrix using scalar multiplication. The resulting 3 × 3 matrix is inserted in place of the corresponding entry in the final matrix. At the end of the first stage, the generated 9 × 9 matrix contains the joint probabilities of the first and second gate's inputs. For example, the entry (3, 3) contains the probability of the first input being in the correct "0" state while the second input is in the correct "1" state (p(I_1 is correct "0") × p(I_2 is correct "1")). It is worth noting that the assumption of independent inputs is a default assumption for all stages. In the second stage, the same multiplication and substitution operations are applied to each entry in the generated 9 × 9 matrix and STPM_3. At the end of this stage, a 27 × 27 matrix is produced, where each entry represents the joint probability of inputs one, two, and three being in the specified states. In the following stages, similar operations are applied to the fourth through N_{inp}th inputs and the produced matrix grows to form the final JPIM of the gate. For additional clarification, we present the JPIM calculation for an arbitrary two-input gate in Figure 5, where the STPMs of the inputs are represented by matrices A and B.

In the second step, we calculate PM = JPIM × PTM. By comparing the result to an ideal matrix (IM), an STPM is constructed. The IM is a $3^{\text{Ninp}} \times 3$ matrix that is computed similarly to the PTM, but all transistors in the gate are assumed to be fault-free. Two types of rows exist in the IM: rows containing a "1" and two "0s" (type 1), and rows containing no "0s" (type 2). To clarify type 1, suppose that the i'th row of the IM contains (1, 0, 0). We can deduce that if the gate is fault-free, then the output must be "0" for the corresponding input vector. The equivalent row in the PM contains (p1, p2, p3) and we perform the following operations: p1 is added to STPM(1, 1), which represents the correct "0" probability; p2 is added to STPM(1, 2), which represents the correct "0" that is transformed to the incorrect "Tri-STATE;" and p3 is added to STPM(1, 3), which is related to the correct "0" that is transformed into the incorrect "1."

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \qquad B = \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix}$$

$$JPIM_{A,B} = \begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 & a_2b_1 & a_2b_2 & a_2b_3 & a_3b_1 & a_3b_2 & a_3b_3 \\ a_1b_4 & a_1b_5 & a_1b_6 & a_2b_4 & a_2b_5 & a_2b_6 & a_3b_4 & a_3b_5 & a_3b_6 \\ a_1b_7 & a_1b_8 & a_1b_9 & a_2b_7 & a_2b_8 & a_2b_9 & a_3b_7 & a_3b_8 & a_3b_9 \\ a_4b_1 & a_4b_2 & a_4b_3 & a_5b_1 & a_5b_2 & a_5b_3 & a_6b_1 & a_6b_2 & a_6b_3 \\ a_4b_4 & a_4b_5 & a_4b_6 & a_5b_4 & a_5b_5 & a_5b_6 & a_6b_4 & a_6b_5 & a_6b_6 \\ a_4b_7 & a_4b_8 & a_4b_9 & a_5b_7 & a_5b_8 & a_5b_9 & a_6b_7 & a_6b_8 & a_6b_9 \\ a_7b_1 & a_7b_2 & a_7b_3 & a_8b_1 & a_8b_2 & a_8b_3 & a_9b_1 & a_9b_2 & a_9b_3 \\ a_7b_4 & a_7b_5 & a_7b_6 & a_8b_4 & a_8b_5 & a_8b_6 & a_9b_4 & a_9b_5 & a_9b_6 \\ a_7b_7 & a_7b_8 & a_7b_9 & a_8b_7 & a_8b_8 & a_8b_9 & a_9b_7 & a_9b_8 & a_9b_9 \end{bmatrix}$$

FIGURE 5 JPIM construction for a two-input gate

The second type is related to generation of the "FLOAT" state in the gate's output when applying the i'th input combination. We interpret the "FLOAT" state as one of the "0," "Tri-STATE," and "1" states with probabilities of p1, p2, and p3, respectively. For inclusion in the STPM, we assume an equally probable expected state for the gate's output. For example, if the "FLOAT" state is interpreted as the "0" state (with a probability of p1), then in the STPM, we select the first column. Therefore, based on the assumption above, we must add p1 to the entry that is located in the first row and first column. For the "Tri-STATE" state, we add p2 to the second row of the second column. Accordingly, for the "1" state, p3 is added to the third row of the third column.

As an example, consider a two-input NAND gate (for additional clarity, the internal circuitry of such a gate is illustrated in figure 9A in Section 3.4), where we assume that the V_{th} values of two P-CNTFETs are equal, as are the V_{th} values of two N-CNTFETs. Therefore, regarding V_{Tri} there are three different regions (R_1, R_2, and R_3). In R_1, V_{Tri} can turn on the N-CNTFETs, but cannot turn on the P-CNTFETs. In R_2, V_{Tri} can turn on both the N-CNTFETs and P-CNTFETs. In R_3, V_{Tri} can turn on the P-CNTFETs, but cannot turn on the N-CNTFETs. As stated previously, to derive an IM, we should assume that all CNTFETs are fault-free, meaning we can extract the IM when V_{Tri} is placed into these three regions according to the gate's PTM calculation methodology. The results are presented in Figure 6.

For multi-fan-in gates, the size of the JPIM increases exponentially. For example, a six-input NAND gate requires a $(3^6)^2 \times (3^6)^2$ JPIM containing 531 441 entries. To reduce the number of required computations, we decompose such a gate into a tree of two-input gates. The STPM derivation procedure is then applied to each sub-circuit. The STPM of the final gate in the tree is considered as the final STPM.

For primitive gates (AND, OR, NAND, and NOR) with N inputs, decomposition is applied based on the following Boolean expressions:

$$I_{N-1} \cdot I_{N-2} \cdot \ldots \cdot I_1 \cdot I_0 = (I_{N-1} \cdot (I_{N-2} \cdot (I_{N-3} \cdot \ldots \cdot (I_1 \cdot I_0) \ldots)))$$

$$I_{N-1} + I_{N-2} + \ldots + I_1 + I_0 = (I_{N-1} + (I_{N-2} + (I_{N-3} + \ldots + (I_1 + I_0) \ldots)))$$

$$\sim (I_{N-1} \cdot I_{N-2} \cdot \ldots \cdot I_1 \cdot I_0) = \sim (I_{N-1} \cdot (I_{N-2} \cdot (I_{N-3} \cdot \ldots \cdot (I_1 \cdot I_0) \ldots)))$$

$$\sim (I_{N-1} + I_{N-2} + \ldots + I_1 + I_0) = \sim (I_{N-1} + (I_{N-2} + (I_{N-3} + \ldots + (I_1 + I_0) \ldots))).$$

The AND, OR, and NOT operators are indicated by ., +, and ~symbols, respectively. A parenthesis on the right-hand side of an expression indicates a two-input gate. In the cases of NAND and NOR gates (the final two expressions), decompositions are performed based on decompositions of equivalent AND and OR gates, respectively. Then, the necessary inversion is accomplished by using a NOT gate following decomposition.

The application of this method for a four-input NAND gate is illustrated in Figure 7. The corresponding tree is presented in Figure 7A. This tree consists of three two-input AND gates and a single NOT gate. This tree is constructed based on the Boolean expression below.

$$\sim(I_3 \cdot I_2 \cdot I_1 \cdot I_0) = \underset{\text{NOT}}{\sim}(I_3 \overset{\text{AND3}}{\cdot} (I_2 \underset{\text{AND2}}{\cdot} (I_1 \overset{\text{AND1}}{\cdot} I_0)))$$

The transistor-level topology of the tree is presented in Figure 7B, where the green CNTFETs are fault-free and T_i is equivalent to T_i in the four-input NAND gate topology (Figure 7C). Additionally, in this figure, for all gates (AND1, AND2, AND3, and NOT), the related transistors are identified using colored regions. As an important point in Figure 7C, the "Tri-STATE" state turns on both the green P-CNTFETs and N-CNTFETs to propagate the faulty state with maximum probability.

$$\begin{array}{c|ccc|}
AB & & & \\
00 & 0 & 0 & 1 \\
0T & 0 & 0 & 1 \\
01 & 0 & 0 & 1 \\
T0 & 0 & 0 & 1 \\
TT & 1 & 0 & 0 \\
T1 & 1 & 0 & 0 \\
10 & 0 & 0 & 1 \\
1T & 1 & 0 & 0 \\
11 & 1 & 0 & 0
\end{array}
\qquad
\begin{array}{c|ccc|}
AB & & & \\
00 & 0 & 0 & 1 \\
0T & 0 & 0 & 1 \\
01 & 0 & 0 & 1 \\
T0 & 0 & 0 & 1 \\
TT & 0 & 1 & 0 \\
T1 & 0 & 1 & 0 \\
10 & 0 & 0 & 1 \\
1T & 0 & 1 & 0 \\
11 & 1 & 0 & 0
\end{array}
\qquad
\begin{array}{c|ccc|}
AB & & & \\
00 & 0 & 0 & 1 \\
0T & 0 & 0 & 1 \\
01 & 0 & 0 & 1 \\
T0 & 0 & 0 & 1 \\
TT & 0 & 0 & 1 \\
T1 & 0 & 0 & 1 \\
10 & 0 & 0 & 1 \\
1T & 0 & 0 & 1 \\
11 & 1 & 0 & 0
\end{array}$$

FIGURE 6 IM for a two-input NAND gate: (A) R_1, (B) R_2, (C) R_3

To verify the accuracy of this decomposition approach, we compared the resulting STPM of the output of the multi-fan-in gate to the STPM of the output of the corresponding decomposed tree. These comparisons were performed for all primitive gates (AND, OR, NAND, and NOR) with fan-in numbers between three to nine, while the STPM for each gate's input was selected randomly to cover all possible cases. We considered the complementary and ratioed design styles in our simulations. In the pass-transistor logic, the core of the design was realized using two connected N-CNTFETs (for additional clarity, the internal circuitry of a two-input NAND gate is presented in figure 10 in Section 3.4), meaning the implementation of high fan-in gates can be accomplished using two-input modules. The problem of a very large JPIM is solved by this design methodology automatically.

The results of our comparisons are presented in Figure 8. Based on these results, we can deduce that the proposed decomposition approach has less than 0.01% error for STPM calculations. Additionally, the problem of exponential growth in JPIM size is resolved by the proposed decomposition method. For example, in the four-input NAND case, each NAND and NOT gate is related to a 9 × 9 and 3 × 3 JPIM, respectively. Therefore, the six-input NAND gate's tree

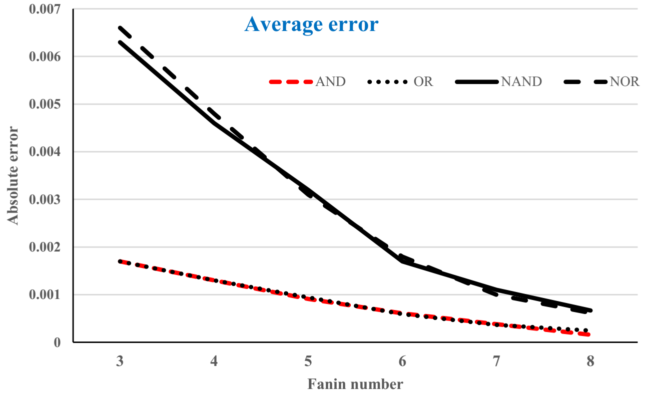


FIGURE 8 Average error of decomposition

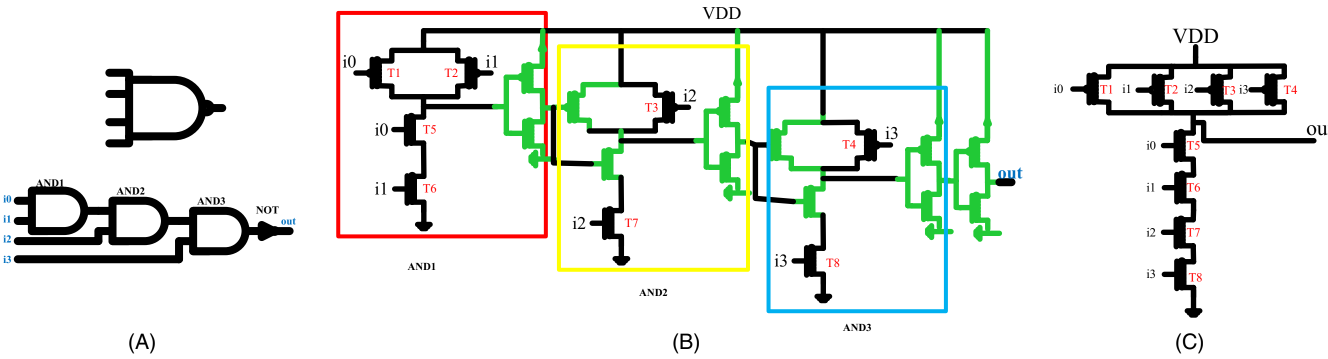


FIGURE 7 Multi-Fan-in NAND gate

consists of five two-input NAND gates and six NOT gates. Consequently, the total number of related JPIM entries is 459 (= 81 × 5 + 9 × 6), which is a significant reduction compared to the 531 441 entries for a six-input NAND gate.

3.4 | Various gate design method examples

In this subsection, we present some examples to clarify the approach of our proposed method for the reliability evaluation of CNTFET-based logic gates. We consider complementary, ratioed, and pass-transistor design methodologies.

In the complementary gate design, the PUN and PDN are implemented using P-CNTFETs and N-CNTFETs, independently. The internal circuitry of a two-input NAND gate is illustrated in Figure 9A. Assume that for the T1 transistor, $|V_{\text{th}}| < V_{\text{Tri}}$, and for the T2 transistor, $|V_{\text{th}}| > V_{\text{Tri}}$. Therefore, the "Tri-STATE" state voltage can turn on T1, but cannot turn on T2. The T3 and T4 transistors (which comprise the PDN) are placed in the same column, meaning they are fully correlated and their threshold voltages are equivalent [25]. We assume that for the T3 and T4 transistors, $V_{\text{th}} < V_{\text{Tri}}$. Then, the PTM_{Comp} and $\text{STPM}_{\text{comp}}$ can be calculated according to the approach described in the previous subsections. For example, suppose that we wish to calculate the second row of the PTM, which is related to A = "0" and B = "Tri-STATE." According to the assumptions outlined above, the input B can turn on T4 and turn off T2, while the input A turns on T1 and turns off T3. The related equations for P_{disc} (up) and P_{disc} (down) are represented in (24) and (25), respectively.

$$P_{\text{disc}}\,(\text{up}) = (1 - \left(1 - P_{\text{O}}\,(T1)\right) \times \left(1 - P_{\text{S}}\,(T2)\right). \quad (24)$$

$$P_{\text{disc}}\,(\text{down}) = 1 - \;P_{\text{S}}\,(T3) \times \left(1 - P_{\text{O}}\,(T4)\right). \quad (25)$$

The internal circuitry of a NAND gate implemented using the ratioed method is illustrated in Figure 9B. Compared to the complementary design, the NAND_{Rat} and $\text{NAND}_{\text{comp}}$ have similar PDN networks, but the PUN in NAND_{Rat} contains only a P-CNTFET, where the gate terminal is connected to the GND (logic state "0") permanently. The related equations for A = "0" and B = "Tri-STATE" are defined in (26) and (27), respectively.

$$P_{\text{disc}}\,(\text{up}) = P_{\text{O}}\,(T1)\,. \quad (26)$$

$$P_{\text{disc}}\,(\text{down}) = 1 - \;P_{\text{S}}\,(T3) \times \left(1 - P_{\text{O}}\,(T4)\right). \quad (27)$$

The internal circuitry of $\text{NAND}_{\text{pass}}$, which is implemented using the pass-transistor design method, is presented in Figure 10. As an example, assume that in the $\text{NAND}_{\text{Pass}}$ topology, the N-CNTFETs are fully correlated with $V_{\text{th}} < V_{\text{Tri}}$. Additionally, suppose that the first NOT gate produces a "Tri-STATE" state in $\overline{B}$, meaning V_{Tri} is connected to the gate terminal of the T2 CNTFET. Subsequently, this transistor is turned on and connects node F to the GND. In contrast, if the T1 CNTFET turns on (as B = "1" or "Tri-STATE," or a short fault in the transistor occurs), then the path from A to F may conflict with the turned on path from F to GND. If the state of A is "0," then two paths connect F to GND simultaneously, meaning the state of F would be "0." If A = "1" or "Tri-STATE," then the first path connects F to the GND, but the second path connects F to V_{dd} or V_{Tri}. These two cases produce the "Tri-STATE" state in the F node.

To compute the PTM of a $\text{NAND}_{\text{pass}}$ gate (Figure 10), three steps are applied for every input vector. First, the STPM of $\overline{B}$ is calculated using the PTM of the corresponding NOT gate and STPM of input B. Second, the STPM of node F is calculated, where T1 and T2 connect node F to input A and the GND, respectively. The turned on/off states of T1 and T2 are determined based on the B and $\overline{B}$ logic values, as well as the $V_{\text{ th}}$ values of the transistors. If A = "0," then there is no possible way to generate "Tri-STATE" or "1" states in the F node. However, if A = "1" (= V_{dd}), then T1 acts as a PUN. In contrast to the complementary and ratioed logic styles, where P-CNTFETs are used in the PUN, this PUN cannot charge the voltage of node F to V_{dd} (that is, $V_{\text{dd}} - V_{\text{th,n}}$).

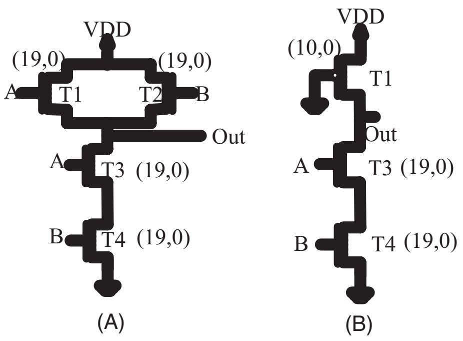


FIGURE 9 (A) $\text{NAND}_{\text{Comp}}$, (B) NAND_{Rat}

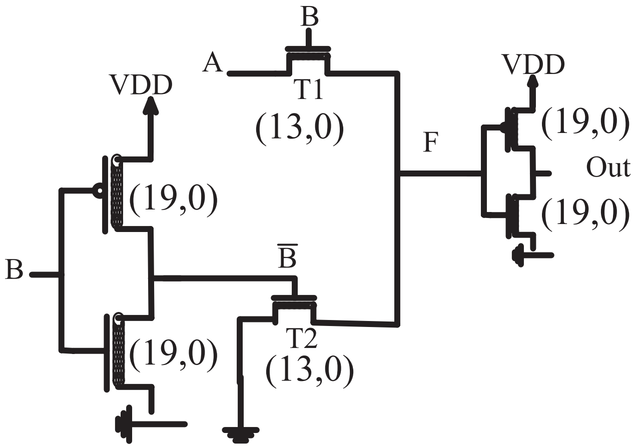


FIGURE 10 $\text{NAND}_{\text{Pass}}$

Suppose that T1 and T2 turn on simultaneously based on the application of proper gate voltages (eg, $V_G(\text{T1}) = V_{\text{dd}}$ and $V_G(\text{T2}) = V_{\text{dd}}$). Then, the generated voltage in node F would be much lower than V_{dd} / 2. In this case, we approximate the logic of node F as the "0" state. Based on the full correlation between CNTs in T1 and T2, short and open faults occur simultaneously. If both transistors contain open faults, then node F will be in the "FLOAT" state. Furthermore, the existence of short faults in T1 and T2 results in a "Tri-STATE" state in node F. In the final step, the STPM of the gate's output is calculated based on the PTM of the second NOT gate and the STPM of node F.

4 | RELIABILITY EVALUATION OF COMBINATIONAL CIRCUITS

The overall flow of the proposed reliability estimation method for combinational logic circuits can be summarized as follows. In the initial step, required parameters such as p_{m}, p_{mr}, and p_{sr} are tuned. Additionally, for each gate type (NAND, NOR, XOR, NOT, etc) the chirality vectors of the gate's CNTFETs are defined. Next, by using (8) and (9), the V_{th} values of all CNTFETs are calculated.

In the next step, the circuit is levelized in topological order. Starting from level 1, where all gate inputs are connected to the primary inputs, the following operations are applied to each gate. Based on the STPMs of the gate's inputs, the JPIM is constructed and the PTM of the gate is calculated using the method discussed in Section 3.1. According to Section 3.3, the STPM of the gate can be derived from the JPIM and the STPMs of the inputs. The calculated STPM is assigned to all branches of the fan-out cone (FOC) originating from the target gate. For the other levels, similar operations are applied until the primary outputs are reached. At this time, all STPMs are calculated and the reliabilities of all circuit nodes can be derived using (22).

One noteworthy issue in this process flow is reconvergent fan-out handling. As stated previously, in the calculation of a gate's JPIM, independency among inputs is assumed, but this assumption is incorrect in the case of reconvergent fan-out. When the branches of an FOC intersect at two inputs of a specific gate, those two inputs become dependent. This error in JPIM computation leads to inaccuracies in the computed STPM of such a gate. Similar to the authors of [28], we use a multiple-iteration approach to solve this problem. In this approach, FOCs that generate reconvergent points are first identified. Next, in each iteration, only one state of the identified FOCs is used in the reliability evaluation flow. For example, if two identified FOCs (F_1 and F_2) generate a reconvergent point in a circuit, then in the first iteration, F_1 and F_2 are assumed to be in states $\text{STPM}_1(1, 1)$ and $\text{STPM}_2(1, 1)$, respectively. This means that in the first iteration, we use STPM'$_1$ and STPM'$_2$ (see (28) and (29)) instead of $STPM_1$ and $STPM_2$, respectively.

$$\text{STPM}_1' = \begin{bmatrix} p(s_0(1) \rightarrow s_0(1)) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{28}$$

$$\text{STPM}_1' = \begin{bmatrix} p(s_0(2) \rightarrow s_0(2)) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{29}$$

This approach generates exact results, but it is not scalable for large circuits. For a circuit with $N_{\text{FOC-rec}}$ problematic FOCs, the total number of necessary iterations is $9^{\text{Nfoc-rec}}$, which grows exponentially (approximately 3.48 × 10^9 for iterations $N_{\text{FOC-rec}} = 10$). In this study, we selected a small number of reconvergent FOCs to achieve enhanced accuracy in our reliability evaluation flow. This selection was performed according to a ranking process based on the number of reconvergent points generated by each FOC. In [34], a correlation-coefficient-based approach was developed to handle the reconvergent fan-out problem efficiently.

The proposed method is applicable to combinational circuits, but by using the sequential-to-combinational conversion methodology developed in [35], we can handle sequential CNTFET-based logic circuits appropriately.

The proposed reliability evaluation method has linear computational and space complexity relative to the number of circuit gates. Therefore, the proposed method is scalable to large circuits. Suppose that a logic circuit contains N_g two-input gates (or is converted into such a circuit according to the proposed decomposition procedure) that are ordered in L levels, each of which contains $n_g(l)$ gates. The proposed algorithm traverses the circuit graph in level-by-level fashion. In each level, two processes are executed. First, we compute a PTM for every gate belonging to the current level. If the average effort (multiplication and summation) required to compute this matrix is P_{PTM}, then the corresponding total computational complexity is $P_{\text{PTM}} \times N_{\text{g}}$. Second, we compute an STPM for the gates in level l using the procedure outlined in Section 3. The average effort required for this step is considered to be P_{STPM} and the overall corresponding complexity is $P_{\text{STPM}} \times N_{\text{g}}$.

For greater clarity, assume that a circuit is synthesized using only two-input NAND gates (we refer to the inputs as A and B). It should be noted that the NAND gate is a universal gate and every switching function can be realized using only NAND gates. In this case, there are two parallel paths in the PUN and a single series path in the PDN. Based on the existence of two P-CNTFETs in the PUN and two N-CNTFETs in PDN, there are five regions for each "Tri-STATE" state in a gate's inputs. Therefore, there are 49 different cases ((AB, number of different cases) = (00, 1), (01, 1), (0T, 5), (T0, 5), (TT, 25), (T1, 5), (10, 1), (1T, 5), and (11, 5)) for each gate's PTM calculation. For each case, according to (11) to (20), nine multiplications, five subtractions, and three additions are required. Therefore, for complete calculation of the PTM, we must apply 49 × (9 + 5 + 3) arithmetic operations ($P_{PTM} = 833$). In the STPM calculation, we perform 81 multiplications to construct the JPIM (Figure 5). Next, (3 × 9) multiplications and (3 × 8) summations are performed in the JPIM ×PTM calculation. The next steps to derive the STPM require (3 × 9) summations. Finally, the total number of required operations required for STPM computation (P_{SPM}) is 159. Ultimately, for a circuit with N_g two-input NAND gates, the total number of arithmetic operations is 992 × N_g. This result indicates linear growth in computational complexity for the proposed reliability evaluation approach.

Space complexity includes the registration of circuit graph information (k_1 memory words, including the type of gate and its fan-in and fan-out interconnections), the STPM of I_{prim} primary inputs (nine memory words for every input), and STPMs of N_g gates (nine memory words for every gate). The gate PTM and STPM calculations are accomplished in level-by-level fashion, as well as in gate-by-gate fashion within each level. Therefore, the space complexity of this process is based on a 9 × 9 JPIM, 9 × 3 PTM, and 9 × 3 IM, which are independent of the number of circuit gates. Consequently, the space complexity of a circuit containing N_g two-input NAND gates can be calculated according to (30).

$$M = (k_1 + 9) \times N_g + I_{prime} \times 9\,. \tag{30}$$

One can see that the space complexity of our proposed method is also linear relative to the number of circuit gates.

5 | SIMULATION RESULTS

5.1 | Reliability comparisons of various single gates

Various design methodologies for a logic gate can be selected by an IC designer. Each design has its advantages and disadvantages in terms of delay, power dissipation, noise margin, etc [36]. The reliability of emerging technologies such as CNTFETs must be considered during gate design selection. An accurate and straightforward methodology for computing the reliability of a logic gate is the simulation of its behavior using Monte-Carlo SPICE (MC SPICE). In such simulations, various scenarios for fault occurrence in CNTFETs are considered. For each scenario, the output of the target gate for different input vectors is derived using HSPICE simulations and compared to the result of a fault-free gate. For a given input vector, if the faulty and fault-free gates generate different output values, then an error occurrence is registered for the corresponding scenario. Suppose that for an arbitrary logic gate, we consider N_{Fault_Scen} fault scenarios, where N_{inp_vec} input vectors are simulated for each scenario. If the total number of error occurrences is Err_{tot}, then the reliability of the gate can be calculated according to (31).

$$Rel_{gate} = 1 - \frac{Err_{tot}}{N_{Fault\_Scen} \times N_{inp\_vec}}\,. \tag{31}$$

However, this method is very time-consuming and has high computational complexity (N_{Fault_Scen} different HSPICE simulations). In contrast, our proposed method can estimate gate reliability rapidly and accurately. To verify the accuracy of our method, we compared the reliability estimates generated by our method to those generated by the MC SPICE method. Various gate types (NAND, NOR, AND, OR) with multiple fan-in numbers (two to eight fan-in numbers for each gate type) were considered in our simulations. Additionally, complementary, ratioed, and pass-transistor design styles were used in our simulations. The states of all CNTFETs in each gate (fault-free, open fault, and short fault) must be determined prior to HSPICE simulation. We generated the required (at least 10 000) fault scenarios using MATLAB for various P_S and P_O probabilities (in the range of 0.001 to 0.1). For each scenario, the related data were added to the MC SPICE simulation externally. The generated output values (from the HSPICE simulations) were then registered in an output file corresponding to the applied input vectors. Finally, error occurrence cases were identified for the target scenario using MATLAB. The average errors of reliability estimation for the proposed method compared to MC HSPICE simulation are presented in Figure 11. According to the simulation results, the average value of estimation error is only 0.01% for various gate types, demonstrating the high accuracy of our proposed method.

$$STPM_{Ideal} = \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.75 \end{bmatrix}. \tag{32}$$

For reliability comparisons using individual gates, three NAND gates (described in Section 3.4) were considered. We applied the proposed method for the STPM derivation of gate outputs when the NAND inputs were connected to fault–free primary inputs. In this scenario, if there is a fault-free NAND gate, the $STPM_{ideal}$ would be equal to (32). All entries except for $STPM_{ideal}(1, 1)$ and $STPM_{ideal}(3, 3)$ are zero. The entries of $STPM_{ideal}(1, 1)$ (correct “0” state) and $STPM_{ideal}(3, 3)$ (correct “1” state) are equal to 0.25 and 0.75, respectively. We illustrate the values of correct “0” and correct “1” states for a conventional CNTFET, ACCNT_5, and ACCNT_15 in Figures 12, 13, and 14, respectively. The x

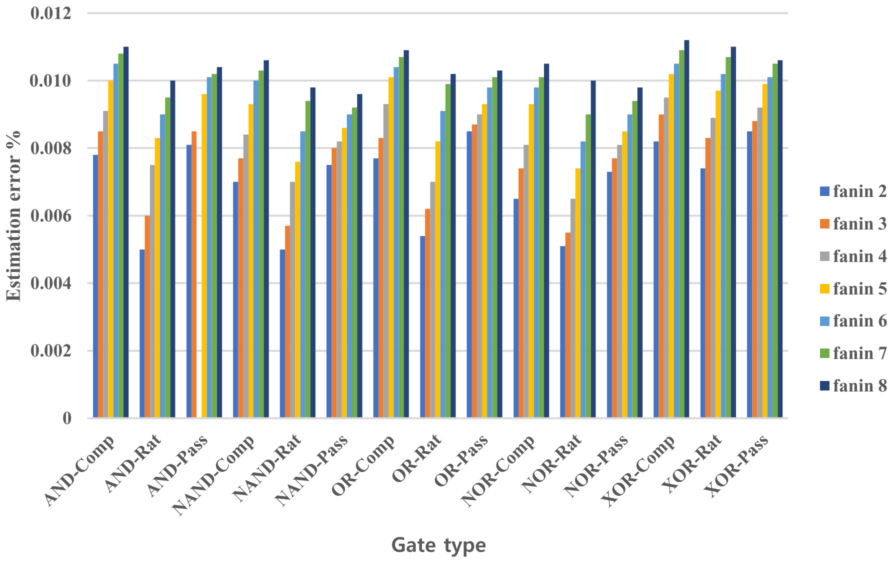


FIGURE 11 Estimation errors for various gate types

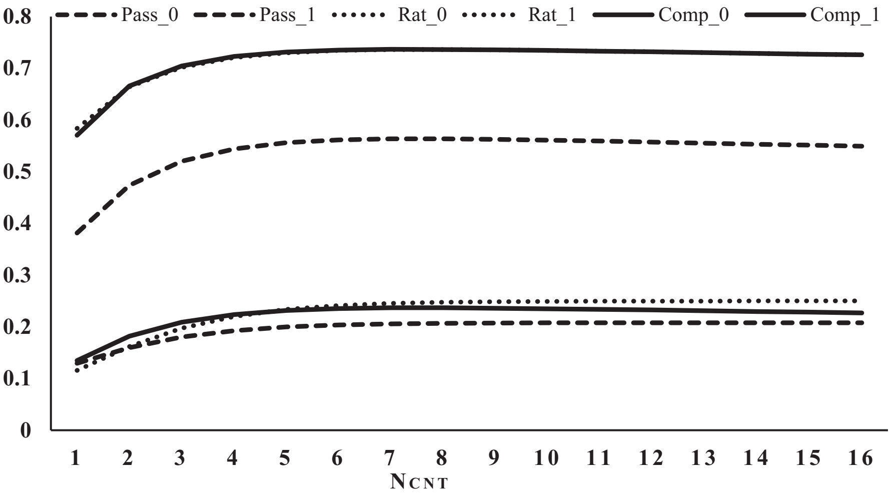


FIGURE 12 Correct “0” and “1” probabilities for a conventional CNTFET

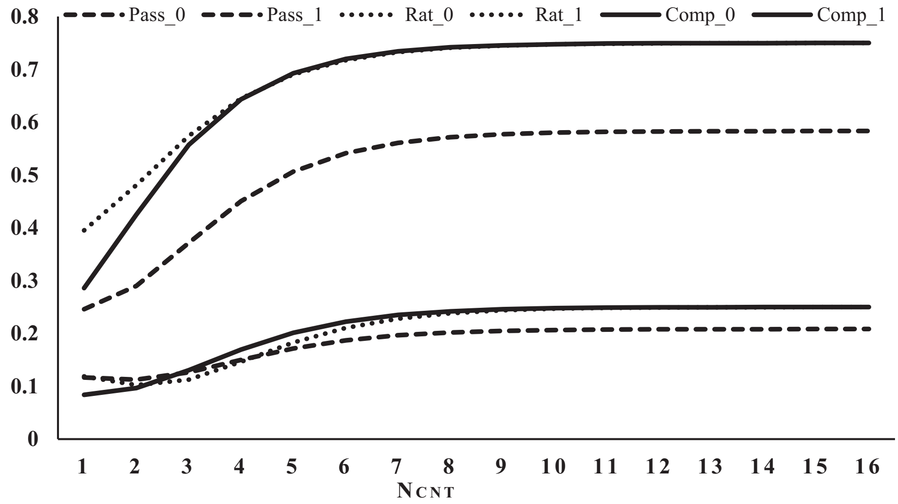


FIGURE 13 Correct “0” and “1” probabilities for ACCNT_5

FIGURE 14 Correct “0” and “1” probabilities for ACCNT_15

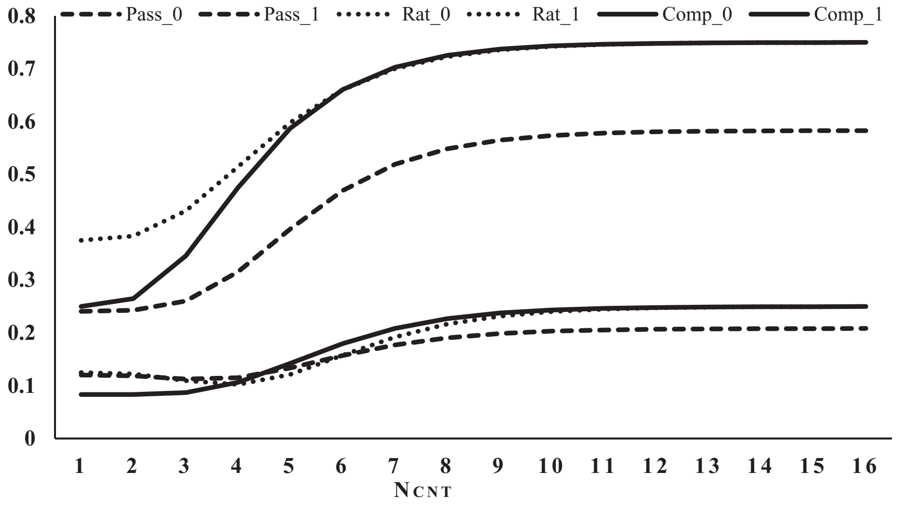
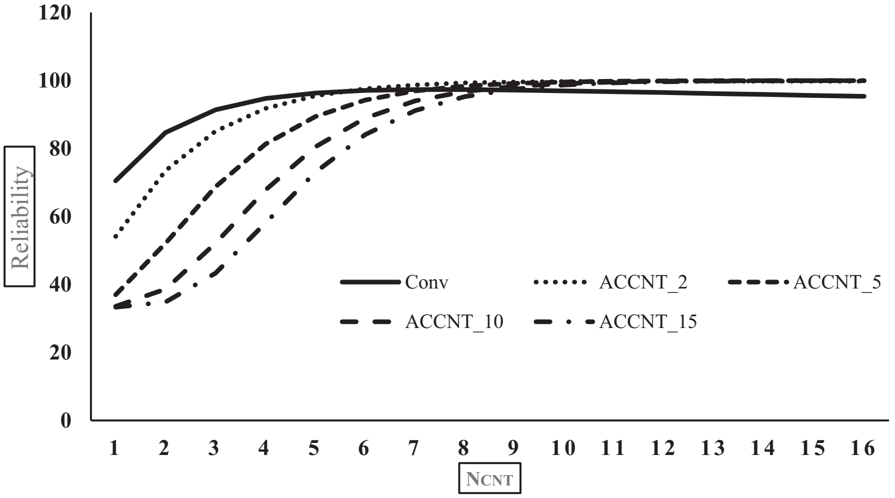


FIGURE 15 Reliability of $\text{NAND}_{\text{Comp}}$



axes in these figures represent the number of CNTs contained in each gate's CNTFETs. For example, $N_{\text{CNT}} = 10$ indicates that there are 10 CNTs in T_1, T_2, T_3, and T_4 in Figure 8A. In the ACCNT_5 and ACCNT_15 gates, any conventional CNTFET is replaced with a row similar to that shown in Figure 2, where 5 and 15 CNTFETs are placed between the D and S nodes, respectively.

In all of these cases, the $\text{NAND}_{\text{Pass}}$ performance is relatively low, meaning the probabilities of correct “0” and “1” states in $\text{STPM}_{\text{Pass}}$ deviate more than the corresponding probabilities in $\text{STPM}_{\text{Comp}}$ and $\text{STPM}_{\text{Rati}}$. The main reasons for this deviation are potential connections from the drain/source terminals of the T1 transistor to the “Tri-STATE” state and the NOT gate, which is placed between node F and the output nodes, resulting in additional erroneous values in the gate's outputs (Figure 10). Therefore, the total number of cases that can lead to “Tri-STATE” state generation in the $\text{NAND}_{\text{Pass}}$ output increases, making the “Tri-STATE” state probabilities greater (see $\text{STMP}_{\text{Pass}}(1, 2)$, and $\text{STMP}_{\text{Pass}}(3, 2)$).

When an ACCNT is used a gate's topology, deviation from the ideal value decreases as the number of CNTFETs in a row increases. This occurs based on the lower short and open fault probabilities of the ACCNT structure. The behavior of NAND_{Rat} and $\text{NAND}_{\text{Compl}}$ differs in producing a correct “1” state. Because the simpler PUN (only one P-CNTFET in NAND_{Rat} compared to two transistors in $\text{NAND}_{\text{Compl}}$) in NAND_{Rat} is always on, this gate can prevent a transition from the correct “1” state to other states more efficiently. Therefore, particularly for small values of N_{CNT}, the correct “1” state probability of NAND_{Rat} is closer to the ideal value (0.75).

The reliabilities of three NAND gates are plotted in Figures 15 to 17. Each figure contains five graphs. In each graph, one type of CNTFET is used to implement the corresponding gate's structure. For example, the graph with the label “*conv*” corresponds to a NAND gate using conventional CNTFETs. The graph labeled “*ACCNT_N*” corresponds to a NAND gate using an ACCNT structure with *N* CNTFETs in a row. For the aforementioned reasons, the reliability of

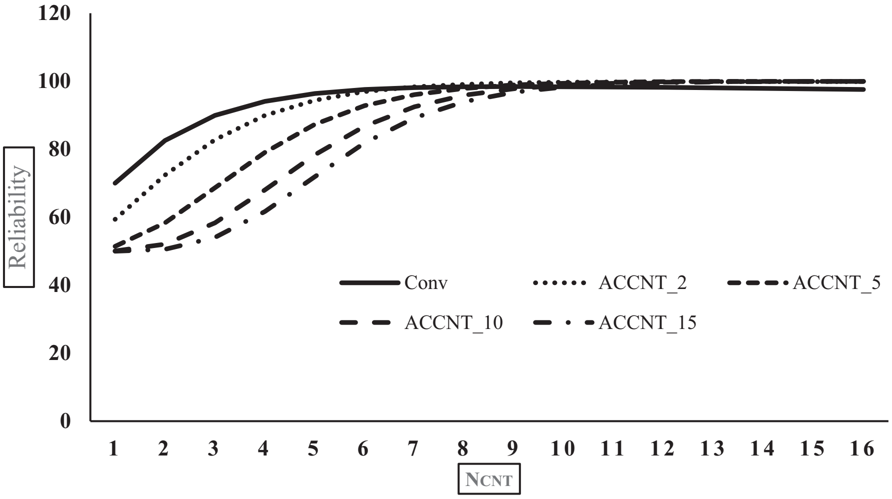


FIGURE 16 Reliability of NAND_{Rat}

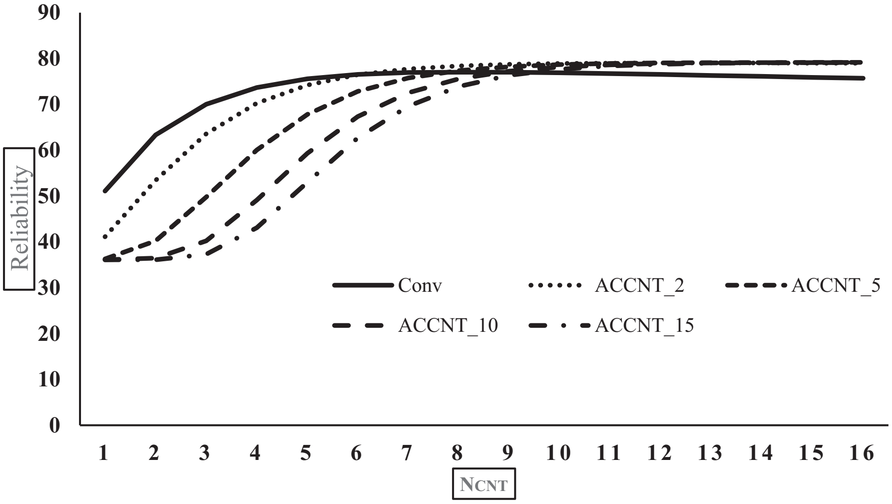


FIGURE 17 Reliability of $\text{NAND}_{\text{Pass}}$

$\text{NAND}_{\text{Pass}}$ is lower than that of the other two architectures. For the interpretation of gate reliability behavior, we divide each plot into two parts. In the lower part ($N_{\text{CNT}} < 8$), P_{O} dominates P_{S}. Therefore, according to these figures, the ACCNT structure with additional transistors is more unreliable. An increasing N_{CNT} leads to a decreasing P_{O}, which results in an improvement in reliability. When P_{O} decreases, P_S increases, which does not change the reliability of the ACCNT-based gates significantly (based on the negligible value of P_S in the ACCNTs). However, for the conventional CNTFET-based NAND gates, P_S has significant value compared to P_{O}. Therefore, in the second part of the graphs, the reliability of the conventional CNTFET-based gate slightly decreases when P_{S} increases.

5.2 | Comparison to previous methods

We compared our proposed method to the three methods developed in [26], [27] and [28]. Hereafter, we refer to these methods as M1, M2, and M3, respectively. The results for all methods were compared using an MC reliability evaluator, where the following steps were applied in each iteration. First, a fault pattern was generated based on the P_{O} and P_{S} values of the CNTFETs. In each fault pattern, the states (normal, short fault, and open fault) of all transistors were determined according to the uniform distribution of the faults in the IC. Next, a subset of all input vectors was randomly selected and the selected vectors were applied to the faulty and fault-free circuits simultaneously. For each applied vector, the logic values of the outputs in the fault-free and faulty circuits were compared. If the logic values did not match, the corresponding error increased by one. One important consideration is how to handle the "FLOAT" state. When the "FLOAT" state appears in the output of a gate, the value is converted to one of the other values with an equal probability (1 / 3). The other major issue is related to the "Tri-STATE" state voltage generated at the output of the gate (V_{Tri}). As mentioned in Subsection 3.1, when both the PUN and PDN are turned on, the drive strength of the turned on paths in the PUN and PDN determines V_{Tri}. For the basic logic gates (AND, OR, etc), to obtain more realistic V_{Tri} values, we simulated (using HSPICE) the possible scenarios that can generate the "Tri-STATE" state at the output of the gate. We recorded the generated V_{Tri} values for all scenarios in a table linked to the gate. Subsequently, if one of the possible "Tri-STATE" states occurred in a gate, the value of V_{Tri} would be calculated based on this table. The next fault pattern was then generated and the steps above were repeated in the next iteration. This approach is very accurate if there are sufficient fault patterns and selected vectors in each iteration. However, achieving acceptable accuracy requires very high run times.

M1 is a semi-analytical method in which each row of the gate's PTM is encoded as a "state sequence" [26]. For example, suppose that we select a sequence size of 1000 and in row r of the gate's PTM, the probabilities of the "0", "Tri-STATE," and "1" states are equal to 0.2, 0.1, and 0.7, respectively. Consequently, there will be 200, 100, and 700 of the "0", "Tri-STATE," and "1" symbols in the sequence, respectively. After generating all PTM-related sequences, a $2^k \times 1$ multiplexer is inserted in place of each gate (k is the gate's input number) and the sequences are connected to the corresponding multiplexer (MUX) inputs. The input pins of the gate are interpreted as selector inputs for the MUX. The randomly generated sequences are assigned to the primary inputs and propagated through the circuit's MUXs. The reliability of a gate is computed by decoding the corresponding MUX's output sequence. The probability of the "0", "1", and "Tri-STATE" states is calculated by dividing the number of related symbols by the sequence length. The reliability of a sequence is computed by summing the "0" and "1" state probabilities. In this method, to achieve high accuracy for reliability estimation, we must adopt a large sequence length that imposes additional run time overhead. Additionally, the decoding process is imperfect based on its interpretation of all "0" ("1") states in the gate's output sequence as correct "0" ("1") states, even though some of these "0" ("1") states may be incorrect. This misinterpretation will result in greater inaccuracy in circuits containing many reconvergent fan-outs. This effect is caused by the existence of many correlated signals in such circuits. If the target sequence of correlated signals contains incorrect "0" ("1") states, then they will likely be maintained through the propagation process.

The second method (M2) is an analytical reliability evaluator in which three states ("0," "1," and "H") are defined for each circuit node. The state "H" represents both the "FLOAT" and "Tri-STATE" states in our method. Regarding a gate's failure probability calculation, three cases are considered: 1) the gate's inputs are fault-free and the gate is faulty, 2) the gate is fault-free and the gate's inputs are faulty, and 3) both the gate and the gate's inputs are faulty. This method operates based on event occurrences in the inputs and outputs of a gate. An error in the output of a gate may occur based on input error propagation through fault-free (case 2) or faulty (case 3) inputs, or through the generation of a faulty output caused by a faulty gate (case 1). This method has some shortcomings. First, it is assumed that the "H" state cannot turn on any CNTFET, which is not true in a practical model. Second, the probability of a gate's output being the "FLOAT" state must be translated into three other states, but in this method, it is considered only as the "H" state. Third, reconvergent fan-outs are not handled in this method.

The M3 method is another analytical method based on PTMs. In this method, all four states ("0," "1," "Tri-STATE," and "FLOAT") are considered for a gate's PTM calculation. One major problem with this method is related to the transformation of a complete PTM (similar to our proposed method's PTM) into a smaller PTM in which "Tri-STATE" state probabilities are eliminated. This transformation makes the reliability evaluation flow similar to gate-level PTM-based analysis, but introduces significant errors based on the exclusion of the "Tri-STATE" and "FLOAT" states in a gate's STPM calculations. Another concern regarding this method is that reconvergent fan-outs are not considered.

As mentioned previously, MC is the reference method used for measuring the accuracy of the other methods. The reliability estimation error for a method M_i was calculated according to (33). We used the ISCAS 85 benchmark circuits to compare the performance of the proposed method to those of MC, M1, M2, and M3.

Table 1 lists the set of ISCAS 85 benchmark circuits and two very large circuits from the ITC 99 benchmark circuits (b18 and b19). The total numbers of CNTFETs included in the benchmark circuits are indicated in second column. The run times (in seconds) and memory usages (in kilobytes) of the reliability evaluation methods are reported in the third and fourth columns of Table 1, respectively. All simulations were executed on a 3.2 GHz microprocessor with 4 GB of

TABLE 1 Run times and memory usages of the proposed method for different benchmark circuits

Circuits	Number of CNTFETs	Runtime (sec)	Memory (Kbytes)
C17	24	6.86	1.48
C432	1006	25.00	36.28
C499	1524	25.39	45.49
C880	2064	25.45	85.01
C1355	2484	28.13	118.05
C1908	4214	31.03	187.95
C2670	6192	37.96	262.69
C3540	8894	49.23	355.57
C5315	13 942	56.02	499.15
C6288	10 112	43.39	511.88
C7552	17 544	63.35	755.37
b18	428 552	1526.77	19 883.25
b19	865 366	3423.79	40 124.74

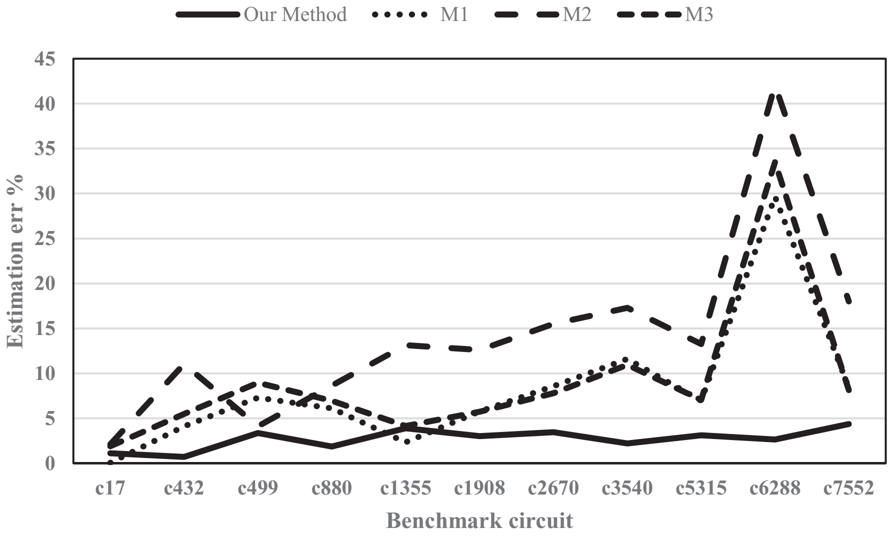


FIGURE 18 Estimation errors for the ISCAS 85 benchmark circuits

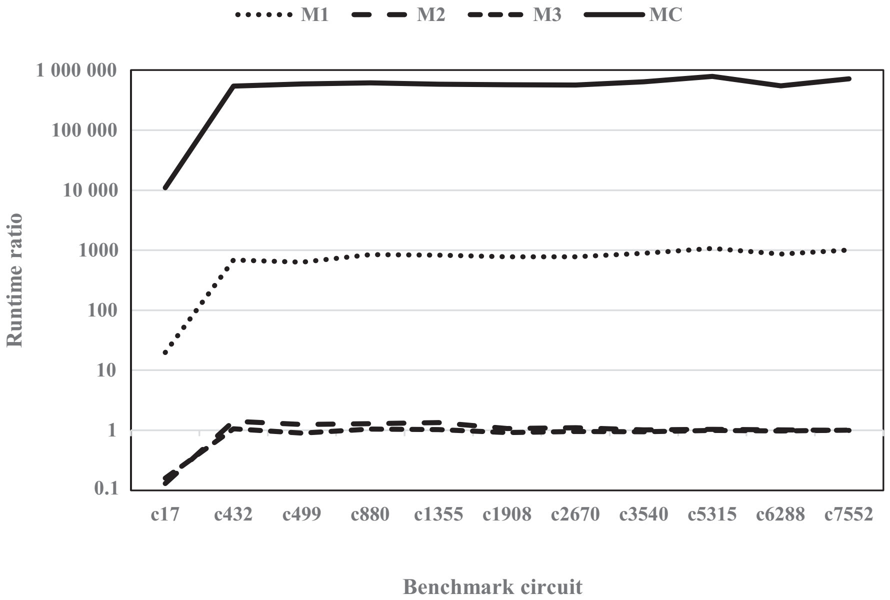


FIGURE 19 Run time ratios of various methods relative to the proposed method

RAM. The results indicate that the proposed method is fast and scalable, even for very large circuits such as b18 (94 249 gates) and b19 (19 0213 gates). It should be noted that for such large circuits, the MC method requires multiple days and several gigabytes of memory to complete its calculations. Therefore, the estimation errors and runtime ratios are not reported for b18 and b19 in Figures 17 and 18.

The average reliability estimation errors for the circuit gates are presented in Figure 17 for the ISCAS 85 benchmark circuits.

$$\mathrm{Err}_i = 100 \times |\mathrm{Rel}_{\mathrm{MC}} - \mathrm{Rel}_{\mathrm{Mi}}| / \mathrm{Rel}_{\mathrm{MC}}. \tag{33}$$

In Figure 18, we present the average results for three values of P_{S} and P_{O} (0.1, 0.01, and 0.001). In all cases, the proposed method outperforms the others methods. The average estimation error of the proposed method is 2.67%, while the average errors of the other methods are 9.08%, 16.21%, and 9.87% for M1, M2, and M3, respectively. M2 exhibits the worst performance because it uses the "H" state instead of the "FLOAT" and "Tri-STATE" states.

In circuits with large numbers of reconvergent fan-outs (C432, C499, and C6288), the estimation error of our method is much lower than those of the other methods. This can be attributed to the inability of M2 and M3 to handle reconvergent fan-outs. Generally, when reducing the transistor error probabilities (P_{O} and P_{S}), the error values become smaller for all methods. For each method, the reliability estimation error increases with increasing circuit size. In all cases, M1 outperforms M2 and M3, which can be attributed to its simulation-based nature and handling of reconvergent fan-outs.

The other important factor is the run times of the methods. The ratios of the runtimes of the other methods relative to the proposed method are presented in Figure 19. Because it is an analytical approach, it is clear that the proposed method is faster than MC and M1 (more than 616 000 times and 837 times faster, respectively). The run times of the proposed method and the other analytical methods (M2 and M3) are of the same order, but M3 requires slightly less time because it performs fewer computations for its PTM and STPM calculations.

6 | CONCLUSION

CNTFETs have emerged as promising candidates to solve the downscaling problems faced by Si-based CMOS technology. In this study, an analytical method was developed for the reliability evaluation of CNTFET-based combinational logic circuits. In the proposed method, a gate's PTM is computed at the transistor level based on the short and open fault probabilities of CNTFETs. According to the gate's topology, the probabilities of generating "0," "1," "Tri-STATE," and "FLOAT" states in its outputs can be calculated. Simulation results demonstrated the accuracy and scalability of the proposed method. For the ISCAS 85 benchmark circuits, our method shows achieved an average reliability estimation error of less than 3%. This error is at least 6% less than the error rates of previous methods. Additionally, the speed-up ratios of the proposed method are more than 6×10^5 times compared to the MC approach and 800 times compared to previous stochastic approaches.

ORCID

Hadi Jahanirad https://orcid.org/0000-0001-8586-6281

REFERENCES

1. A. D. Franklin et al., *Sub-10 nm carbon nanotube transistor*, Nano Lett. **12** (2012), 758-762.
2. S. Qiu et al., *Solution-processing of high-purity semiconducting single-walled carbon nanotubes for electronics devices*, Adv. Mater. **31** (2019), no. 9, 1800750.
3. T. Skotnicki et al., *The end of CMOS scaling: Toward the introduction of new materials and structural changes to improve MOSFET performance*, IEEE Circuits Devices Mag. **21** (2005), 16-26.
4. J. M. Rabaey and S. Malik, *Challenges and solutions for late-and postsilicon design*, IEEE Des. Test Comput. **25** (2008), 296-302.
5. M. K. Q. Jooq et al., *Design and performance analysis of wrap-gate CNTFET-based ring oscillators for IoT applications*, Integration. **70** (2020), 116-125.
6. S. Fujita et al., *Circuit and systems based on advanced MRAM for near future computing applications*, in Proc. Symp. VLSI Circuits, (Kyoto, Japan), June 2019, pp. C278-C279.
7. H. Jahanirad, *Efficient reliability evaluation of combinational and sequential logic circuits*, J. Comput. Electron. **18** (2019), 343-355.
8. Z. Yao, C. L. Kane, and C. Dekker, *High-field electrical transport in single-wall carbon nanotubes*, Phys. Rev. Lett. **84** (2000), 2941-2944.
9. T. Durkop et al., *Extraordinary mobility in semiconducting carbon nanotubes*, Nano Lett. **4** (2004), 35-39.
10. A. Javey et al., *Carbon nanotube field-effect transistors with integrated ohmic contacts and high-k gate dielectrics*, Nano Lett. **4** (2004), 447-450.
11. I. A. Khan and N. Alam, *CNTFET based circuit design for improved performance*, in Proc. Int. Conf. Electr., Electron. Comput. Eng. (Aligarh, India), Nov. 2019, pp. 1-5.
12. B. Ghavami and M. Raji, *Failure characterization of carbon nanotube FETs under process variations: Technology scaling issues*, IEEE Trans. Device Mater. Reliab. **16** (2016), 164-171.
13. S. K. Vendra and M. Chrzanowska-Jeske, *Tube redundancy in statistical evaluation of critical path delay of CNFET circuits in the presence of tube variations*, in Proc. IEEE Int. Conf. Nanotechnol. (Macao, China), July 2019, pp. 374-377.
14. E. Abiri, A. Darabi, and S. Salem, *Design of multiple-valued logic gates using gate-diffusion input for image processing applications*, Comput. Electr. Eng. **69** (2018), 142-157.
15. K. Tamersit, *Computational study of p-n carbon nanotube tunnel field-effect transistor*, IEEE Trans. Electron. Devices. **67** (2020), 704-710.
16. M. Gholipour and N. Masoumi, *Design investigation of nano electronic circuits using crossbar based nano architectures*, Microelectron. J. **44** (2013), 190-200.
17. A. Bachtold et al., *Logic circuits with carbon nanotube transistors*, Sci. **294** (2001), 1317-1320.
18. S. J. Han et al., *High-speed logic integrated circuits with solution-processed self-assembled carbon nanotubes*, Nat. Nanotechnol. **12** (2017), 861-866.
19. C. Wang et al., *Device study, chemical doping, and logic circuits based on transferred aligned single-walled carbon nanotubes*, Appl. Phys. Lett. **93** (2008), 33101.
20. P. Zarkesh-Ha and A. A. M. Shahi, *Stochastic analysis and design guidelines for CNFETs in gigascale integrated systems*, IEEE Trans. Electron. Devices. **58** (2011), 530-539.
21. J. Zhang, N. P. Patil, and S. Mitra, *Probabilistic analysis and design of metallic-carbon-nanotube-tolerant digital logic circuits*, IEEE Trans. Comput. Aid. D. **28** (2009), 1307-1320.
22. R. Ashraf, M. Chrzanowska-Jeske, and S. G. Narendra, *Functional yield estimation of carbon nanotube-based logic gates in the presence of defects*, IEEE Trans. Nanotechnol. **9** (2010), 687-700.
23. S. Banerjee, A. Chaudhuri, and K. Chakrabarty, *Analysis of the impact of process variations and manufacturing defects on the performance of carbon-nanotube FETs.*, IEEE Trans. Very Large Scale Integration Syst. **28** (2020), no. 6, 1513-1526.
24. A. Lin et al., *ACCNT—A metallic-CNT-tolerant design methodology for carbon-nanotube VLSI: Concepts and experimental demonstration*, IEEE Trans. Electron Devices. **56** (2009), 2969-2978.
25. S. Lin, Y. B. Kim, and F. Lombardi, *CNTFET-based design of ternary logic gates and arithmetic circuits*, IEEE Trans. Nanotechnol. **10** (2011), 217-225.
26. J. Liang et al., *Design and reliability analysis of multiple valued logic gates using carbon nanotube FETs*, in Proc. IEEE/ACM Int. Symp. Nanoscale Archit. (Amsterdam, Netherlands), July 2012, pp. 131-138.
27. F. Saeidi, B. Ghavami, and M. Raji, *A fast method for process reliability analysis of CNFET-based digital integrated circuits*, J. Comp. Elect. **17** (2018), 571-579.
28. B. Srinivasu and K. Sridharan, *A transistor-level probabilistic approach for reliability analysis of arithmetic circuits with applications to emerging technologies*, IEEE Trans. Reliability. **66** (2017), 440-457.

29. S. J. Tans, A. R. M. Verschueren, and C Dekker, *Room temperature transistor based on a single carbon nanotube*, Nat. **393** (1998), 49-52.
30. C. G. Almudever and A. Rubio, *Variability and reliability analysis of CNFET technology: Impact of manufacturing imperfections*, Micro. Reliab. **55** (2015), 358-366.
31. B. Ghavami et al., *Statistical functional yield estimation and enhancement of CNFET-based VLSI circuits*, IEEE Trans. VLSI. **21** (2013), 887-900.
32. F. Yang et al., *Chirality pure carbon nanotubes: Growth, sorting, and characterization*, Chem. Rev. **120** (2020), 2693-2758.
33. M. Ahmad and S. R. P. Silva, *Low temperature growth of carbon nanotubes—A review*, Carbon. **158** (2019), 24-44.
34. H. Jahanirad, *CC-SPRA: Correlation coefficients approach for signal probability-based reliability analysis*, IEEE Trans. Very Large Scale Integr. Syst. **27** (2019), 927-939.
35. H. Jahanirad and K. Mohammadi, *Sequential logic circuits reliability analysis*, J. Circuits Syst. Comput. **21** (2012), no. 5, 1250040.
36. M. A. Savari and H. Jahanirad, *NN-SSTA: A deep neural network approach for statistical static timing analysis*, Expert Syst. Appl. **149** (2020), 113309.

AUTHOR BIOGRAPHIES

Hadi Jahanirad received his BS degree in Electrical Engineering from the Department of Electrical Engineering, Khaje Nasir Toosi University, Tehran, Iran in 2006, and his MS degree and PhD from the Iran University of Science and Technology, Tehran, Iran in 2008 and 2012, respectively. Since 2013, he has worked with the Department of Electrical Engineering at the University of Kurdistan, Sanandaj, Iran, where he is currently an assistant professor. His main research interests include digital system design, VLSI design, reliability analysis of logic circuits, digital circuit testing, approximate computing, and evolutionary computing.

Mostafa Hosseini received his BS degree in Electrical Engineering from the Department of Electrical Engineering, Islamic Azad University, Hamedan Branch, Hemedan, Iran in 2016 and his MS degree in Electrical Engineering from the Department of Electrical Engineering, University of Kurdistan, Sanandaj, Iran in 2018. His main research interests include VLSI design, fault-tolerant systems, digital circuit testing, approximate computing, and the reliability analysis of logic circuits.