Efficient Delay Fault Characterization of Resistive Open Defects in Standard Cells Using Resistive Fault Dominance

Gowsika Dharmaraj*, Abhijit Chatterjee*, Adit Singh** and Arani Sinha+
*School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA.
*Department of Electrical and Computer Engineering, Auburn University, Auburn, USA.
+Intel Corporation, USA

***Abstract*—Stringent quality requirements for safety-critical applications drive the demand for "zero defects" in modern ICs. In this context, delay characterization of standard cells for resistive open defects is an increasing concern due to aggressive timing margins in digital circuits. The problem is made worse by the large number of open defect sites in standard cells, combined with a wide range of defect resistance values for each site. This incurs possible prohibitive costs for defect simulation and characterization. To alleviate this complexity, we propose Resistive Fault Dominance (RFD) for resistive open defects. RFD eliminates simulations of certain open defects with intermediate defect resistance values that are guaranteed to exceed specified timing margins for standard cells, based on tests for specific "dominant" open defects. This can significantly reduce the computational costs of cell library characterization and simulation effort by 84%-91%. An algorithmic fault simulation methodology for resistive open defects on parasitic-extracted (PEX) transistor-level netlist is developed.**

***Index Terms*—Open defects, Delay fault testing, Defect characterization, Critical faults, Resistive fault dominance, Elmore Delay.**

I. Introduction

Cell-aware Testing (CAT) methods target internal open-circuit and short-circuit defects within standard cell libraries [1,2]. Standard cells (also called cells in this paper), which are the building blocks of ICs, comprise a network of PMOS and NMOS transistors and implement certain Boolean functions. During manufacturing, these suffer from resistive open and short defects (or faults, we use both interchangeably in the paper) that can alter their logic function and delay properties. CAT-based test pattern generation for defect detection is implemented in two steps: (1) defect coverage characterization of single and two-pattern tests for defects injected into the cell library, and (2) hierarchical circuit-level automatic test pattern generation (ATPG) for application of cell-level tests from the circuit inputs/scan-based test logic. The tests identified in (1) are justified to the circuit inputs and scan flip-flops in (2). Defect characterization for identifying high defect coverage (multi-pattern) tests for a complete cell library is expensive and time-consuming, even though it is a one-time setup cost [5]. While CAT reduces defect levels significantly, library characterization for defects demands exhaustive analog fault simulation across all defect sites for all defect resistance values, increasing fault simulation and characterization complexity (the broad CAT methodology generally simulates only extreme values of open and short defects). Depending on the standard cell, the location of the defects and the defect resistance values, certain intermediate open/short defects (we focus only on the open defects in this work) cause marginal timing failures [7] and necessitate more than one time-frame tests [6]. A resistive open defect is defined by a 2-element list [defect location (cell netlist node n with open defect), defect resistance value R] = $[n, R]$. Also, a maximum allowable timing margin D (or slack) is allocated for each standard cell. Any cell propagation delay larger than D due to a resistive open defect is flagged as a delay fault, the implication is that tests which detect the defect are included in the test pattern sequences for that cell. The problem (2) above is not within the scope of this work, but rather, User Defined Fault Model (UDFM) or cell-level tests are developed, which can be used to synthesize circuit-level tests using state-of-the-art test generation tools.

> The core problem is the characterization/grading of the coverage of resistive open defects (defined by $[n, R]$), for specified two-pattern (2-pattern) tests, with the *minimum amount* of SPICE simulation effort. Generally, this is difficult because of the large range of defect resistance values R for each open defect at node n to be evaluated for impact on cell delay. The number of defect simulations needed to assess defect coverage of test sequences is minimized using the concept of *Resistive Fault Dominance* (RFD). This reduces defect simulation effort by 84%-91% over exhaustive fault simulation of all open defects with diverse defect resistance values. Additionally, this helps identify effective test sequences for standard cells.

RFD proposed here is different from the traditional fault collapsing technique, the fault dominance for stuck-at-faults [21]. In RFD, certain resistive open defects are not simulated as they are either guaranteed to cause delay violations or do not cause violations, as discussed later.

If an open fault is present in the pull-up network (PUN) of a cell, and the output is inverting, then it affects the low-to-high propagation delay (or rising transition delay) of the cell (t_{pLH}). Conversely, if it is in the pull-down network (PDN), then it affects the high-to-low propagation delay (or falling transition delay) of the cell (t_{pHL}), as shown in Figure $1(b)$.

Key contributions: The key contribution of the work is introducing a novel understanding of the behavior of resistive open defects at transistor terminals that cause timing violations using *Elmore Delay* analysis, which drives the idea behind RFD for defects in serial transistor chains located in cells.

Resistive Fault Dominance (RFD): Consider there are N possible fault sites and R values taken for fault simulation. If a test pattern for a defect $[n_i, R_i]$ at i_{th} location indicates that the timing margin D for a standard cell is exceeded, then it is a violation. If from this result we can deduce a different defect $[n_j, R_j]$ at j_{th} location will also violate timing, without explicitly simulating the latter defect, then we say that the defect $[n_i, R_i]$ dominates the defect $[n_j, R_j]$.

For a serial chain of NMOS (PMOS) transistors (we call it a *branch/path* in the paper), connected between a node and ground (GND) or power supply (VDD), we specifically investigate and show that:

1) Monotonic trends in the rising or falling transition delay of cell output are observed when the same resistive valued open fault advances from a transistor terminal connected to the GND (or VDD) towards the node present at the opposite end of the chain, usually the cell output node.
2) The largest cell delay is observed when a resistive drain/source/gate open is at the terminal of the NMOS (PMOS) transistor in the branch considered, connected to GND (VDD). Conversely, the smallest delay is observed when the same defect is present at the terminal of the transistor, connected to the cell output node.
3) Based upon the observations above, only certain resistive open defects are simulated, and RFD is used to make deductions about whether other defects in serial chains of transistors in the path violate or satisfy delay specifications. Thereby, eliminating the need to simulate all possible faults. A systematic procedure for minimizing defect simulations using RFD is developed, where we only choose specific transistors and certain resistance values for fault injection. An algorithmic 2-pattern UDFM test generation methodology is implemented to target the chosen faults.

For proof of concept, we use discrete defect resistance (R) values between the minimum ($1K\Omega$) and maximum ($700K\Omega$), based upon data of likely open defect resistance values obtained from industry. It is seen that the overall procedure reduces defect simulation complexity, allowing speedups in characterization estimation of standard cell libraries by more than 9X. Those defects that escape 2-pattern tests generated by the algorithm can subsequently be subjected to exhaustive 2-pattern or 3-pattern tests for enhanced coverage as described in [3], [6].

The rest of the paper is organized as follows: Section II discusses the prior works related to the field. Section III presents the analysis of drain/source open faults, gate open faults, the fault simulation methodology and the fault characterization problem. Section IV covers the topic of tapered designs. The implementation details and algorithms developed are covered in Section V. Finally, Section VI presents experimental results, followed by the conclusion and the future work.

II. Prior Work

Over recent years, various static and dynamic fault modelling schemes [8-15] have been developed to support Automatic Test Pattern Generation (ATPG) architectures. Cell-Aware Testing (CAT) [1, 2], a defect-based approach, targets cell-internal defects by applying test patterns to the layout-extracted netlist to expose bridging and open faults. While effective, CAT incurs significant setup costs, longer test times, and inflated test pattern counts, particularly when targeting the weakest internal faults (WF-CAT) [16]. Despite these efforts, [17] highlights the inefficiency of existing methods in detecting all open defects.

Several optimization methodologies address these challenges. Embedded Deterministic Test Point technology [18] reduces CAT pattern counts by resolving internal signal conflicts, increasing faults targeted per pattern. The circuit simulation time is minimized in [19] by narrowing the defect set to essential faults. A current-based Switch-Level ATPG (SL-ATPG) [20] improves pattern generation by detecting short-circuit current paths. A systematic approach in [3] reduces exhaustive simulations for library characterization using 2 and 3-pattern timing tests. A defect-equivalence based flow in [5] collapses fault sets, streamlining library characterization and reducing simulation via the Defect-Detection Matrix (DDM). Logical undetectability, addressed in [22] using the Undetectability Checker Algorithm (UCA), further reduces simulation times. Recent work on timing-aware ATPG (TA-ATPG) [23] limits runtime and pattern counts by focusing on relevant faults, leveraging fault-filtering schemes and path timing slack. To the best of our knowledge, no prior work specifically involves any study on the behavior of defects on cell delay, which paves the way for analysis on the netlist for efficient defect characterization. We propose an algorithmic approach to reduce the simulation overhead of exhaustive open faults efficiently, particularly those that impact circuit timing. The following section provides a detailed examination of the effect of opens using the RC delay model and the Elmore delay approximations.

III. Key Concepts and Approach Overview

A. RC Model and Elmore Delay of FET chains to analyze drain/source opens

A complementary CMOS-based cell has a Pull-Up-Network (PUN) comprising PMOS transistors connected to the supply and a Pull-Down-Network (PDN) of NMOS transistors connected to the ground, implementing a Boolean function. The output is made high by charging the output capacitance

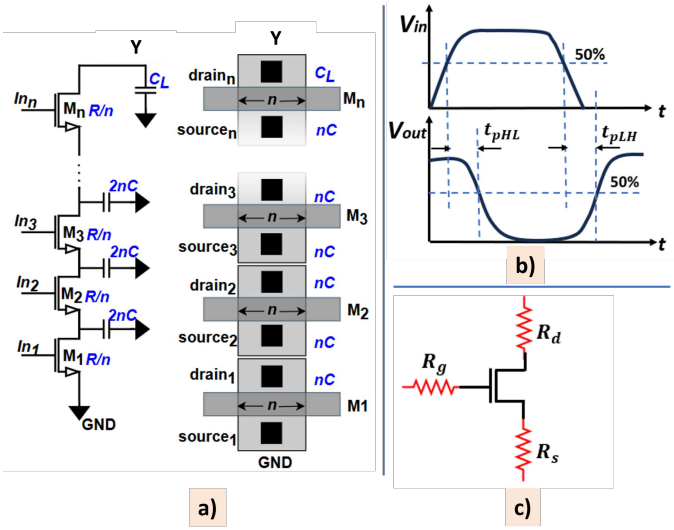


Fig. 1: a) Uniformly-sized NMOS chain and its equivalent parasitic capacitances, for isolated source and drain diffusions b) Propagation delay of signal c) Open fault injection sites

through PMOS transistors and low by discharging it through NMOS transistors. We assume that the PDN and PUN are sized for equal driving strengths for derivation purposes in this section. RC delay models approximate the nonlinear behavior of transistors by using an average resistance and capacitance over the gate's switching range. While this method has limitations in capturing detailed analog behavior, it provides a reliable approximation for delay estimation [4], which is our focus in this work. Let's first consider a chain of n equally sized NMOS transistors and assume R and C are the unit effective resistance and capacitance (a similar analysis can be extended to PMOS transistors). The width of a transistor in the chain is n times the unit width of a minimum-sized NMOS of the inverter. The effective resistance is thus R/n, as the resistance is inversely proportional to its width, and the diffusion capacitance at the source or the drain is nC since the capacitance is directly proportional to the width. To examine the delay effects of the opens, we consider each source and drain having their isolated region of contacted diffusions, as shown in Figure $1(a)$. The generalized fall time of an RC tree using Elmore Delay [24] is given by:

$$t_{fall} = \sum_{i=1}^{n} R_i C_i \tag{1}$$

where C_i is the diffusion capacitance at node i and R_i is the effective resistance, i.e., the sum of all the resistances on the shared path. Expanding this for the circuit in Figure $1(a)$ gives the fall time equation of the fault-free PDN:

$$t_{fall} = \sum_{k=1}^{n-1} \frac{kR}{n}(2nC) + n\frac{R}{n}(C_L) \tag{2}$$

Now, we observe the fall times by injecting opens at the drain/source of the transistor from M_1 to M_n (see Figure 2).

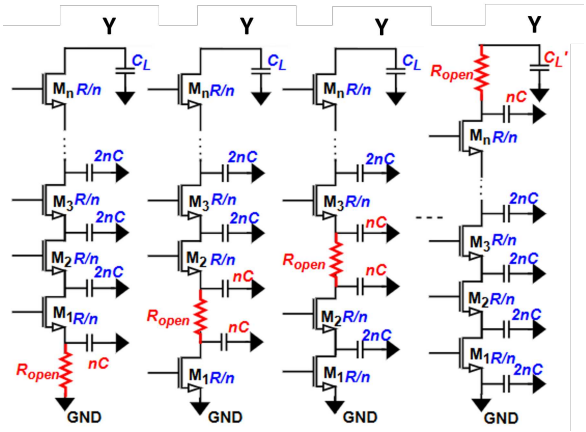


Fig. 2: Injection of open faults at drain and source terminals from M_1 to M_n along with their effective resistances and capacitances.

When present at the drain of M_1, we have:

$$t_{fall} = R_{open}(nC) + \sum_{k=1}^{n-1}(R_{open} + \frac{kR}{n})(2nC) + (R_{open} + n\frac{R}{n})C_L \tag{3}$$

Similarly, when present between M_1 and M_2, the fall time is given by:

$$t_{fall} = n\frac{R}{n}C + (n\frac{R}{n}C + (n\frac{R}{n} + R_{open})nC) + \sum_{k=2}^{n-1}\left(R_{open} + \frac{kR}{n}\right)(2nC) + \left(R_{open} + n\frac{R}{n}\right)C_L \tag{4}$$

Table I shows the fall time equations for four cases of injecting opens as in Figure 2 along fault-free PDN, where C_L is the load capacitance of the fault-free circuit and C_L' is the modified load capacitance when open is at the drain of M_n. Depending on the location of the open, we can see that the number of R_{open} appearances or additions to the fault-free equation when advancing from the footer to the output, decreases and thereby decreasing the fall time.

Fault Location	**Fall Time Equation** (T_{fall})
Fault-free PDN	$\sum_{k=1}^{n-1}\frac{kR}{n}(2nC) + RC_L$
Open Between M_n and Y	$\sum_{k=1}^{n-1}\left(2nC\cdot\left(\sum_{j=1}^{k} j\frac{R}{n}\right)\right) + \sum_{k=1}^{n} nC \cdot \left(k\frac{R}{n}\right) + \sum_{k=1}^{n}\left(R_{\text{open}} + \frac{kR}{n}\right)(C_L')$
Open Between M_2 and M_3	$6RC + nR_{\text{open}}C + \sum_{k=3}^{n-1}\left(R_{\text{open}} + \frac{kR}{n}\right)(2nC) + (R_{\text{open}} + R)C_L$
Open Between M_1 and M_2	$2RC + nR_{\text{open}}C + \sum_{k=2}^{n-1}\left(R_{\text{open}} + \frac{kR}{n}\right)(2nC) + (R_{\text{open}} + R)C_L$
Open at M_1	$nR_{\text{open}}C + \sum_{k=1}^{n-1}\left(R_{\text{open}} + \frac{kR}{n}\right)(2nC) + (R_{\text{open}} + R)C_L$

TABLE I: Fall Time Equations of Fault-Free and Faulty PDN

Let's take an example of a 3-input NAND (NAND3) gate to clarify the trend. The NAND3 standard cell is sized to balance

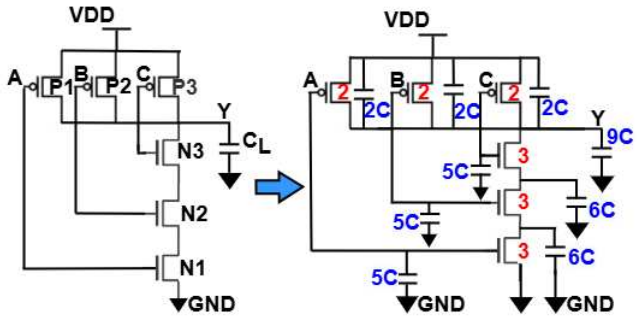


Fig. 3: Equivalent NAND3 gate along with effective widths and switching capacitances

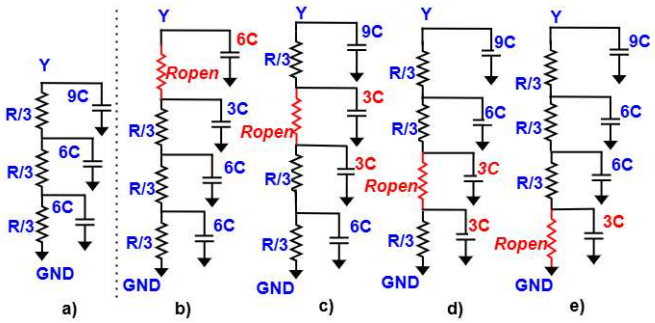


Fig. 4: a) Fault-free RC model of PDN of NAND3 cell having isolated diffusion regions b) Fault between N3 and Y c) Fault between N2 and N3 d) Fault between N1 and N2 e) Fault between N1 and GND

the driving strengths of PUN and PDN. When sizing to the minimum-sized inverter, the NMOS of the NAND3 gate is sized three times the NMOS of the inverter and the PMOS is the same size as the PMOS of the inverter. Let *C* be the effective unit capacitance. The gate capacitances are the sum of the PMOS and the NMOS gate capacitances, *5C(=2C+3C)* and the output capacitance is *9C(=2C+2C+2C+3C)*, as shown in Figure 3 (showing only switching capacitances). Considering each source and drain having its isolated region of contacted diffusions, the equivalent parasitic capacitance of two series-connected transistors is *6C(=3C+3C)*. Figure 4 shows the fault-free and faulty NAND3 cases and the equivalent RC tree of the PDN for the respective cases. The fall-time equation derived using Elmore delay is shown in Table II. As the

Fault Location	**Fall Time Equation** (T_{fall})
Fault-free PDN	$15RC$
Open Between $N3$ and Y	$15RC + 6R_{open}C$
Open Between $N2$ and $N3$	$15RC + 12R_{open}C$
Open Between $N1$ and $N2$	$15RC + 18R_{open}C$
Open at Footer	$15RC + 24R_{open}C$

TABLE II: Fall time equations of NAND3 gate for all five cases in Figure 4

fault injected is moved from the footer toward the output on that particular branch, it is evident that the fall time decreases. Because the fall time depends on the number of R_{open} additions to the fault-free equation, the same trend applies to cells with unequal drive strengths and when the transistors are not uniformly sized, as in tapered designs. The cell behavior for falling and rising transition propagation delays follows the fall and rise time trends of the cell's output node and is validated through simulations. The simulation results in Section VI corroborate this trend for cells with tapered designs and unequal drive strengths.

Based on the above analysis, if there are two faults α and β (both exceeding the timing margin D of the cell) defined by $[n_i, R_i]$, $[n_j, R_j]$ and let D^α, D^β be the delay of the cell in the presence of α and β respectively in two defect locations i and j in a branch.

$$\alpha \text{ dominates } \beta \implies D^\alpha > D^\beta \tag{5}$$

For a gate to suffer a delay failure, we assume that the delay of the gate is increased by 20% due to a resistive open defect. This delay failure threshold, however, can be specified by the user. We consider two cases

Case A: Consider first a resistive open defect β given by [n_i, R_j] that is present at the transistor terminal that is connected to the terminal of the PDN (PUN) of the gate *closest to the gate output*. If β *violates* the delay failure threshold for the gate, then from Elmore delay analysis of the serial PDN (PUN) chain of transistors, it is seen that any open defect α given by [n_k, R_l] for $R_l \geq R_j$ at any other transistor source or drain terminal in the chain will also violate the delay failure threshold for the gate. In other words, $D^\alpha \geq D^\beta$ as in Equation 1 and fault α *dominates* fault β. In such a case, we can drop all such faults α from further fault simulation.

Case B:Now consider a resistive open defect β given by [n_i, R_j] that is present at the transistor terminal that is connected to the GND terminal of the PDN of the gate (or VDD for PUN). If β *does not violate* the delay failure threshold for the gate, then from Elmore delay analysis of the serial PDN (PUN) chain of transistors, it is seen that any open defect α given by [n_k, R_l] for $R_l \leq R_j$ at any other transistor source or drain terminal in the chain will also not violate the delay failure threshold for the gate. In other words, $D^\beta \geq D^\alpha$ and fault β *dominates* fault α. In such a case, we can drop all faults α from further fault simulation.

In our approach, we calculate a critical open defect resistance for Case A which is defined to be the *minimum* resistance value R_j which violates the delay threshold for the gate. This is then used to determine the corresponding faults α that can be dropped from fault simulation. For Case B, a critical open defect resistance value defined to be the *maximum* resistance value R_j which does not violate the delay threshold for the gate. This is used to determine the corresponding faults α that can be dropped from further fault simulation.

B. Analyzing Transistor Gate Opens

The transistor gate opens exhibit similar behavior as observed with the drain/source opens for both uniformly sized

Fig. 5: Flowchart

and tapered designs. For a PDN (PUN), the gate open at the transistor close to the node connecting PUN and PDN creates the smallest delay, and the gate open at the transistor close to GND (VDD), the largest.

Consider the same PDN in Figure $1(a)$. Assume that the output node is initially charged to VDD, and the input to the transistor at the footer creates the latest transition to VDD. The load capacitance will be charged to VDD, and the parasitic capacitances have some initial charge. So, discharge of the precharged output node happens directly. Although at the beginning of the discharge cycle, M_n strongly influences the current, this is only for a short duration until C_L is discharged to a certain value. After that, M_1 is the dominant transistor, through which current sinks to the GND. The input signal to M_1 is the critical signal influencing the cell delay. When the open is at the gate of M_1, and also because the input to this gate makes the latest transition, the parasitic capacitance and C_L do not discharge until M_1 is turned on, thus the rate of discharge of all these capacitances is significantly increased. However, when an open is present at the gate of M_n, the parasitic capacitances are discharged well before the complete discharge of C_L. Regardless of which input signal makes the latest transition in the chain, M_1 will be the critical transistor that causes the critical delay of the path.

Therefore, the gate open at M_n will create the smallest and the open fault at M_1, the largest falling transition delay of the cell output in that branch. For the PUN, if an open is at the gate of the transistor close to VDD, this creates the largest rising transition delay of the cell output compared to the gate open at the transistor at the opposite end of that branch. Thus, by using RFD and identifying the *critical resistances* in a branch, we can eliminate the range of resistances and defect locations in that branch, which is guaranteed to exceed or meet the desired timing margin D.

C. Overview of the methodology

The flow diagram is shown in Figure 5. We identify the fault-free, worst-case output node rise and fall delay by applying the appropriate 2-pattern tests. These serve as a reference for comparison with timing values obtained from faulty circuits. Then, a graph is constructed from the transistor-level netlist. Next, we have to find all the nodes which connect the PUN and PDN in a cell as follows:

1) Basic Standard cells: For simple inverting cells like NAND, NOR, XOR, XNOR, AOI, etc, output is the only connecting node (Y as seen in Figure $6(a)$).

2) Complex cells: For non-inverting and other complex cells, there may be several connecting nodes, before the output

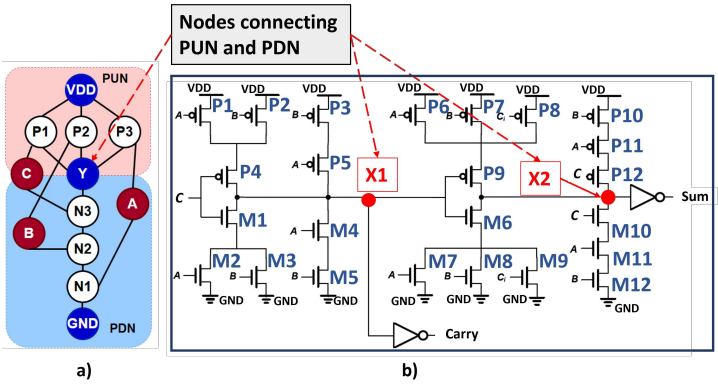


Fig. 6: a) Graph of NAND3 cell and b) A CMOS full adder showing their connecting nodes

node. For the full adder in Figure $6(b)$, there are two such nodes $X1$ and $X2$.

The next step is choosing the transistors in the PUN and PDN whose terminals are connected to the VDD, GND and the connecting node. Opens faults are injected at those transistor terminals by using a modified binary search. Finally, we find the smallest resistance (*critical resistance*) that fails to meet the timing for each fault site. So, using RFD:

- In a branch, for drain/source/gate opens, we eliminate the resistance values greater than the *critical resistance* of the transistor closer to the connecting node, when present at any other location, knowing that it will exceed the specified delay margin D of the cell. Similarly, we eliminate the resistance values less than the *critical resistance* of the one closer to VDD/GND when present at any other location, knowing that it will not exceed D.

This is shown and explained with example scenarios as seen in Figures 7 and 8. It is important to note here that RFD is applied on a per-pattern basis and eliminates open resistive faults from explicit simulation for any/every test pattern chosen for targeting faults. RFD does not limit the composition of multi-pattern tests simulated for complete cell characterization. We only reduce the faults simulated which can be detected by the same test pattern. Hence, no test patterns are dropped in any way that might achieve additional coverage of resistive defects when applied to the standard cell from the inputs of the circuit in which it is embedded.

Using the cell-level test generation in this work, of all the faults that can be detected by the same pattern, we simulate only the dominant ones and eliminate all others for simulation, since we know that the fault will satisfy one of the conditions (1) or (2) mentioned in Section III(A). *Therefore, even for*

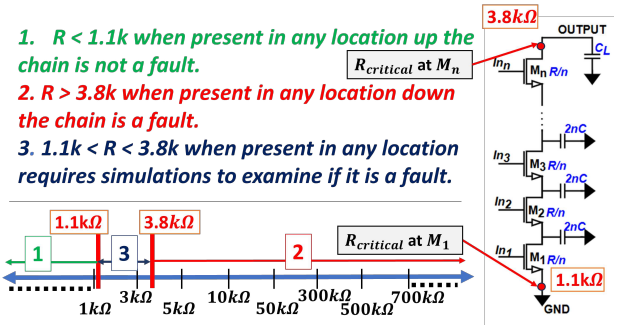


Fig. 7: An example showing the range of values which are faulty and not faulty for a basic PDN for drain/source opens

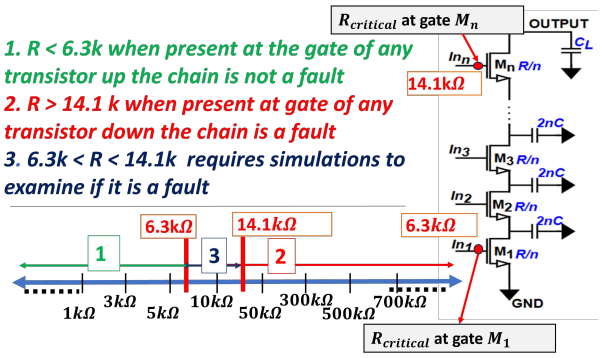


Fig. 8: An example showing the range of values which are faulty and not faulty for a basic PDN for gate opens

the faults that aren't considered for simulation, we have corresponding test patterns available for fault characterization. The entire methodology is automated, which requires the fault-free netlist of the cell (`Netlist`) and the corresponding test bench file (`TB`). The implementation is explained using the algorithms in Section V.

When we say choose transistors in the PUN and PDN, whose terminals are connected to the VDD, GND and the connecting node, this implicitly means that we find *all* the possible *worst-case single* (for example, see Figure 11) paths for every connecting nodes in a cell and we apply RFD within all those branches. This may induce complexity for larger cells, as there can be numerous worst-case paths in both PUN and PDN. However, this concept of dominance can also be extended to be applied to only *few* branches, as opposed to applying it to *all* the branches. Although this is not the scope of the current work, this reduces the characterization problem further. In brief, this can be achieved as follows: if there are more than three single worst-case branches, then we can choose two branches, the branch that creates the largest and the smallest delay, then apply RFD to the chosen fault sites in those two branches only. This gives the minimum and maximum *critical resistances* for these two respective branches, for chosen fault sites. Henceforth, this helps to reduce fault sites and R values in the other branches without explicit simulations. By applying RFD between the branches, the test patterns can also be reduced further, and those that target the worst timing violation can be prioritized.

D. Fault characterization problem

In CAT, the defect matrix M for a cell is constructed during the initial library characterization by performing exhaustive analog simulations. This is a binary matrix, whose columns are the possible faults in a netlist, and rows are the input patterns. This matrix's entry $M(i, j)$ is 1 if the input pattern i detects a particular fault j and 0 otherwise. We consider partial open faults, which are modelled as resistors of some resistance values. We assume a model, where each of the three terminals of the transistor can have an open a shown in Figure $1(c)$. These partial open are the delay faults, which create timing issues, rather than any faulty logic at the output. We choose eight R values for our simulation. These are 1kΩ, 3kΩ, 5kΩ, 10kΩ, 50kΩ, 300kΩ, 500kΩ, 700kΩ) which span a range of values, that are common in general. For an n-input cell, with m number of transistors, there can be $3 \times m$ fault sites. Each of the fault sites can have R values, leading to $3 \times R \times m = 3mR$ faults. The number of 2-pattern tests that result in only one input switching in the second time frame will be $n2^n$ [3]. The number of exhaustive simulations required for characterizing this one cell using 2-pattern tests would be $n2^n \times 3mR$. The time complexity to perform this would be $O(2^{2n} \times mR)$. This implies that as the number of inputs n increases, the number of SPICE simulations grows exponentially, making it computationally hard (time and resource-consuming) for large values of n. However, using RFD, we reduce this complexity by reducing the m and R using minimal test patterns for simulations, as explained in Sections V and VI.

IV. Tapered Designs

Tapering of channel widths in serially connected transistor chains is gradually reducing the widths from the footer to the top of that chain [4], [25], [26]. Tapering increases the performance of the CMOS circuit by minimizing the propagation delay, power consumption and area. Commonly used tapering shapes are linear and exponential [23]. In linear tapering, widths of the adjacent transistors decrease by a constant factor, whereas in exponential tapering, the ratio of widths of the adjacent transistors is fixed and yields more high-speed circuits than in linear tapering [25]. In this work, since the focus is on delay, we design and perform simulations on exponentially tapered domino circuits (such as 4-input NAND and a few OAI cells with many inputs), having a fixed tapering factor α in the range $0 < \alpha \leq 1$ (Category 1), with small output capacitance [25]. Modelling a tapered serial chain is crucial to capture the nonlinearities of the transistors [22]. However, since speed is the only criterion of concern, a linear resistive model of the transistors in the serial chain is used to observe the effect of open defects. To analyze drain/source opens, let's begin with Elmore delay analysis on a fault-free, serially connected NMOS chain (a similar analysis can be extended to the PMOS chain). Since effective resistances are inversely proportional to transistor widths, we have $R_1 < R_2 < R_3 < ... < R_n$ as seen in Figure 10, where C'_L is the modified load capacitance when

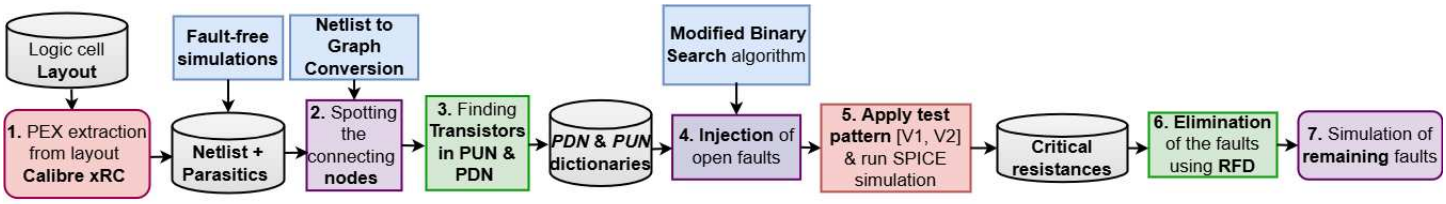


Fig. 9: Implementation flow of RFD

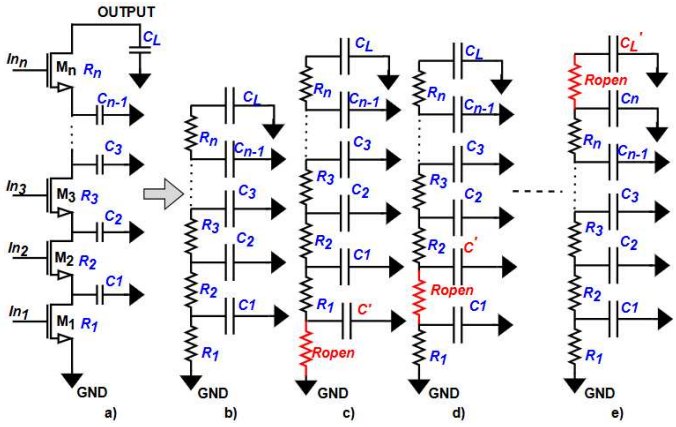


Fig. 10: a) Serially connected NMOS chain having widths of $M_1 > M_2 > M_2 > ... > M_n$ b) Fault-free RC model c) Open at M_1 d) Open between M_1 and M_2 e) Open between M_n and Y

open is at the drain of M_n. The fall time of the chain using the Elmore Delay formula is given by:

$$t_{fall} = \sum_{i=1}^{n-1} \left(C_i \cdot \left(\sum_{j=1}^{i} R_j \right) \right) + C_L \cdot \sum_{j=1}^{n} R_j \qquad (6)$$

The equation (6) can be used to derive fall/rise times for tapered designs by injecting opens. For exponentially tapered designs, if the width of the bottommost transistor is *W*, the succeeding transistor up the chain will be αW, the next one with $\alpha^2 W$ and so on. Modelled effective resistances from the bottom will be R/W, $R/\alpha W$, $R/\alpha^2 W$ and so on, assuming *R* as the unit effective resistance of the untapered transistor. Effective capacitances from the bottom of the chain by considering the isolated source and drain diffusions will be $C(1+\alpha)$, $C\alpha(1+\alpha)$, $C\alpha^2(1+\alpha)$ and so on, assuming *C* as the unit effective capacitance of the untapered transistor. Thus, the fall time equations for fault-free and faulty PDN can be derived similarly to the ones shown in Table I. We will observe that the fall time and thus the falling transition delay increase when the open defect advances from M_n to M_1 because the number of additions of R_{open} to the fault-free equation increases. The same behavior of the gate opens for uniform sizing is observed for the tapered designs. The gate open at M_n will create the smallest delay and M_1 the largest, for the same reason as explained in Section III (B).

Algorithm 1: Identifying transistors for fault injection

Input: `Netlist, TB`
Output: Transistors for fault injection (*PUN* & *PDN*)
Perform fault-free simulation;
Identify worst-case rising and falling cell delay;
Construct graph G of the netlist;
Create list N, the nodes connecting PUN & PDN;
Create lists $T1$ and $T2$, listing transistors in PUN & PDN;
foreach *tran in* $T1$ **do**
  if *tran terminal connected to "VDD or any* $n \in N$*"* **then**
    PUN['*VDD*'].append($tran$);
    PUN[n].append($tran$);
foreach *tran in* $T2$ **do**
  if *tran terminal connected to "GND or any* $n \in N$*"* **then**
    PDN['GND'].append($tran$);
    PDN[n].append($tran$);

So, even for the tapered designs, the aforementioned RFD algorithm can be applied, and thus we can eliminate a significant number of faults, corroborated by the simulation results.

V. Implementation

Implementation flow is shown in Figure 9. Algorithm 1 covers Steps $1-3$. We first perform fault-free simulations of the cell that give falling and rising delay values for all possible 2-pattern tests of the cell. This is followed by graph generation (G) and finding the connecting nodes (N) between the PUN and PDN. Two lists, $T1$ and $T2$ are created, which list transistors in the PUN and PDN, respectively. Note that all transistors in $T1$ will be PMOS type, and those in $T2$ will be NMOS type. From each of the lists $T1$ & $T2$, the transistors, in the presence of an open fault at their drain/source/gate terminal, creating the maximum and minimum cell delays are found. From Section III, these are the ones whose terminal is connected to any nodes: *VDD*, *GND*, any $n \in N$. We create two dictionaries *PUN* and *PDN* that have information on the transistors for fault injection, along with the node they are connected to (VDD, GND and for all $n \in N$).

Using Algorithm 2, the *critical resistances* are determined. For each of the transistors listed in *PUN* and *PDN*, the terminal (drain/source/gate) that is connected to its associated node (VDD/GND/n) is identified. The open fault of a certain resistance value is then injected into this terminal (Step 4), and we assume only one fault is injected at any time in any fault

Algorithm 2: Identifying Critical Resistances

Input: `Netlist`, `TB`, *PUN, PDN, R* values
Output: CR, the critical resistances
begin
  foreach *node* n *in PUN and PDN* **do**
    foreach *transistor* T *in node* n **do**
      Find ON path that has T with the largest Elmore Delay;
      Choose vector V_2 such that:;
        Gates of transistors in the ON path are ON, all others OFF;
      Choose vector V_1 such that:;
        $[V_1, V_2]$ creates worst-case falling/rising delay & is Boolean satisfiable;
      while $R < 20\%$ *ideal falling/rising delay* **do**
          Inject R at the terminal of T connected to n such that:;
            R is chosen by modified binary search;
          Run Simulation;
      $CR[T]$.extend($[R, [V1, V2]]$);

site. ON-path is the path from VDD to n (for faulty PMOS) or from n to GND (for faulty NMOS). All the transistors in that path must be turned ON. The OFF paths are the paths that are parallel to the ON path and whose transistors must be turned OFF. We find an appropriate two-pattern test $[V1, V2]$ to initialize the output node, activate and propagate the fault to the output in reverse chronological order (Step 5). Firstly, we find vector $V2$, which turns ON the transistors in the ON path, that charges or discharges the output capacitance through the faulty transistor. If there is more than a single path through the faulty transistor, choose the worst-case, which charges or discharges more parasitic capacitances, creating the largest delay among all other paths. $V1$ should create an opposite transition at the output compared to $V2$, which initializes the output node to a known logic value. Various possible initialization patterns of $V1$ can create different $[V1, V2]$ pairs for a rising or a falling transition at the output for a single $V2$. We find $V1$, such that $[V1, V2]$ pair that creates the *maximum* cell delay. We find the maximum because this will give the smallest possible resistance that can exceed the ideal margin D. Sometimes, the chosen input vector may not produce the smallest resistance value and/or cannot be realized at the circuit/block level inputs (when the cell is embedded in a large circuit), which can be found by backtracking using PODEM [21]. In those cases, we choose the $[V1, V2]$ pair that causes the next-largest delay and which can be realized at the primary inputs to the circuit. This is applicable to target any drain, source, or gate opens.

The next step is the determination of the *critical resistances*, CR. This stores the corresponding test pattern used for that fault, along with the resistance value. We define a range of values, say M values in total (M is the same for all the cells). We used 100Ω to $200K\Omega$ with a step size of 10, from which the resistances to be injected are chosen based on a modified

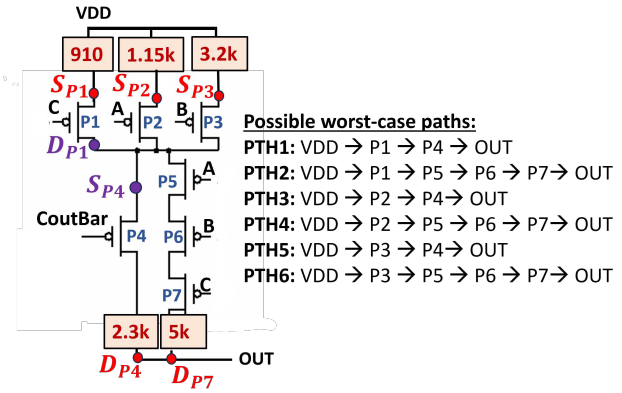


Fig. 11: An example PUN showing critical resistances (in Ω) at the chosen fault sites and a list of worst-case charging paths when OUT is charged through a single path from VDD

binary search algorithm. The search stops when it finds the smallest resistance that exceeds 20% of the ideal rising/falling cell delay at that particular fault site, which is the *critical resistance* for that transistor terminal. In Step 6, we eliminate faults (both fault sites and R values) based on RFD.

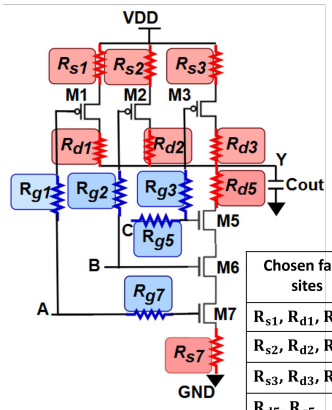
For a network like that in Figure 11, it is clear that multiple single worst-case paths can be taken through the same fault site. For example, how do we eliminate R values for the drain of $P1$ (D_{P1}), where there are two maximum *critical resistances* ($2.3k\Omega$ and $5k\Omega$) and one minimum (910Ω) associated with 2 paths (PTH1 & PTH2)? Any of the two can be taken to charge OUT through the same $P1$, and the path taken is input vector-dependent. Since we eliminate any R greater than the critical resistance of the terminal connected to any $n \in N$ when present up in the chain, we will consider the smallest critical resistance, i.e., it will be $2.3k\Omega$ and not $5k\Omega$. So, we thus eliminate all $R > 2.3k\Omega$ and $R < 910\Omega$ for the fault site D_{P1}. Consider another case: eliminating faults for S_{P4}. Depending on the input vector, any of the 3 paths: PTH1, PTH3, and PTH5 can be taken to charge OUT. Here, there are 3 minimum critical values (910Ω, $1.15k\Omega$ and $3.2k\Omega$) and one maximum value ($2.3k\Omega$). Since we eliminate any R less than the minimum critical resistance of the terminal connected to VDD when present down in the chain, we choose the smallest of all, i.e., it will be 910Ω and not $1.15k\Omega$ or $3.2k\Omega$. A similar analysis is performed for all the other fault sites with two or more *critical resistances* associated with each possible path taken through that particular fault site. Finally, step 7 involves simulating the remaining faults from the defined list of 8, R values between the two extreme values found for that branch. We show simulation results for various cells in detail in the next section.

VI. Experimental Results

The experiments were performed on standard cells and a few circuits designed using NCSU 45nm FreePDK technology [28]. We considered 15 cells, consisting of standard cells with and without equal drive strengths, a CMOS full adder and 3 different tapered cell designs as seen in Table III. A supply of $0.8V$ was used, and Synopsys HSPICE tool was employed.

Cell	# MOSFETS	Total # Two-Pattern tests for switching cell output	Total # fault sites	Total # Faults	# MOSFETS chosen	# Fault Sites chosen	# Faults eliminated for explicit simulation	# Test Patterns Used	% Decrease in faults considered for simulation	Total Run Time (in min)
NAND3X1	6	14	18	144	5	13	121	4	84.03%	2.15
OR2X2	6	6	18	144	6	16	126	3	87.50%	3.31
NOR3X1	6	14	18	144	5	13	122	4	84.72%	2.12
AND2X1	6	6	18	144	5	16	128	3	88.89%	3.57
AND2X2	6	6	18	144	5	16	130	3	90.28%	3.33
XOR2X1	12	8	36	288	12	28	252	4	87.50%	5.41
XOR2X2	12	8	36	288	12	28	256	4	88.89%	5.16
AOI21X1	6	30	18	144	6	13	122	4	84.72%	2.16
XNOR2X1	12	8	36	288	12	28	254	4	88.19%	5.83
OAI21X1	6	30	18	144	6	13	130	4	90.28%	2.35
OAI22X1	6	30	18	144	6	13	134	4	93.06%	2.38
CMOS FULL ADDER	28	32	84	672	23	31	611	12	90.92%	6.98
Tapered Domino NAND4	6	4	18	144	3	7	133	2	92.36%	1.16
Tapered Domino OA654321	25	1440	75	600	10	23	548	9	91.33%	5.44
Tapered Domino OA424242	22	1024	66	528	8	19	481	7	91.10%	4.42

TABLE III: Fault analysis data obtained using RFD for various logic cells



Two-pattern tests for charging Y	Two-pattern tests for discharging Y
[1,1,1]->[0,0,0]	[0,0,0] -> [1,1,1]
[1,1,1]->[0,0,1]	[0,0,1] -> [1,1,1]
[1,1,1]->[0,1,0]	[0,1,0] -> [1,1,1]
[1,1,1]->[0,1,1]	**[0,1,1] -> [1,1,1]**
[1,1,1]->[1,0,0]	[1,0,0] -> [1,1,1]
[1,1,1]->[1,0,1]	[1,0,1] -> [1,1,1]
[1,1,1]->[1,1,0]	[1,1,0] -> [1,1,1]

Chosen fault sites	Critical resistance in kΩ	Test pattern used
R_{s1}, R_{d1}, R_{g1}	3.58, 14.36, 35.82	[1,1,1]->[0,1,1]
R_{s2}, R_{d2}, R_{g2}	1.69, 8.06, 23.42	[1,1,1]->[1,0,1]
R_{s3}, R_{d3}, R_{g3}	0.92, 5.01, 16.29	[1,1,1]-> [1,1,0]
R_{d5}, R_{g5}	5.3, 20.69	[0,1,1]->[1,1,1]
R_{s7}, R_{g7}	0.42, 6.92	[0,1,1]->[1,1,1]

Fig. 12: Fault sites for a 3-input NAND gate, exhaustive two-patterns and critical resistances corresponding to the fault sites

A test bench like in [27] with two cascaded inverters for each primary input of the cell and 4 cascaded inverters as the load in the primary output is used to emulate a real circuit operation. Table III presents various logic cells and their fault analysis. It shows cell-specific data such as the number of MOSFETs, the total number of exhaustive two pattern tests that ***only result in output logic switching in the second time frame***, total fault sites, number of faults (# fault sites $\times$ # R values), number of MOSFETs chosen for fault injection for RFD, number of faults eliminated using RFD for simulation, the number of test patterns used for our simulation using RFD, % decrease in faults to be explicitly simulated and finally the total run time for finding *critical resistances* using modified binary search for the chosen faults sites. We observe that the run time for each cell is proportional to the number of fault sites and the convergence time to find the *critical resistances* using the binary search. The total time for all 15 cells was 55.78 minutes. If the full open fault set were considered, the total simulation time would be 542.36 minutes (approx. 9.04 hours) for all 15 cells. The speed-up in simulation time achieved is 9.5X when using RFD, against simulating all faults. Consider the NAND3 gate in Figure 12, which can have 144 faults. There are 4 worst-case single paths (combining both PUN and PDN) that can charge and discharge the connecting node, which is the cell output itself. RFD is applied to all 4 branches, considering one fault at a time. Figure 12 shows the 13 chosen fault injection sites out of 18, and the bottom table shows the *critical resistances* determined. The table at the top shows that 4 out of 14 test patterns were used to target these faults. We make the following conclusions based on the table in Figure 12, for our simulation setting. Note that for any transistor terminal for which the *critical resistance* was found, we can eliminate all the 8 R values since we know that any $R >$ *critical resistance* will exceed the margin D and $R <$ *critical resistance* will not exceed D. Firstly, let's take PDN, the branch of $M3$, $M4$, $M5$. **Drain/Source opens:** $R > R_{d5}(5.3k\Omega)$ will fail when present at any other drain/source terminal of any transistor down the chain, and any $R < R_{s7}(420\Omega)$ will pass. So, the number of drain/source simulations eliminated is $(8{+}5{+}5{+}8 = 26)$. **Gate opens:** $R < R_{g5}(6.9k\Omega)$ will pass and $R > R_{g7}(20.69k\Omega)$ will fail. So, the number of gate simulations eliminated is 23 $(8+7+8 = 23)$. Let's take a branch in PUN for analysis, the branch of $M1$. **Drain/Source:** Any $R > R_{d1}(14.36k\Omega)$ will fail and $R < R_{s1}(3.58k\Omega)$ will pass. So, the number of drain/source simulations eliminated is 16 **Gate:** $R > R_{g1}(35.82k\Omega)$ will

fail and $R < R_{g1}(35.82k\Omega)$ will pass. So, 8 simulations are eliminated. A similar analysis can be extended to the remaining 2 branches in the PUN of the cell. Thus, we eliminated $121/144$ faults in the NAND3X1 gate, leading to an 84.03% reduction in total faults to be simulated. The remaining 23 faults must be simulated. The same analysis is extended to other cells as shown in Table III. Domino circuits NAND4, OAI654321 and OAI424242 (assuming each NMOS gate has a unique input) with tapering widths were taken from [25]. The transistor widths were sized using α =0.9, and the ratio of load capacitance to the parasitic capacitance was kept < 1. We can infer that for many input gates, having more depth serial chains like OAI654321 and OAI424242 cells, using RFD, we eliminate faults that need not be simulated by more than 90%, by applying very few patterns. ***These numbers depend on the R values considered, simulation settings and circuit environment***. Three major factors, the depth of serial chain networks, the number of such chains, and the size of the circuit, have a significant impact. The higher the three, the more faults can be eliminated. We can benefit more from using RFD as the circuit gets more complex with many transistors. Moreover, for large and complex circuits, we can reduce a lot of simulation effort and defect characterization time by applying RFD to a few single worst-case branches as discussed in Section III(C).

VII. Conclusions

This work shows an algorithmic way of analyzing the netlist by choosing the desired fault injection sites and identifying the minimum resistance failing to meet the timing, reducing a significant number of faults and simulation time. The results demonstrate the scalability and effectiveness of RFD for fault analysis in complex circuits. Such methods offer a promising direction for enhancing fault detection while minimizing computational overhead. As a future work, we plan to perform hierarchical circuit-level APTG test generation, with the defect matrix obtained using RFD for fault characterization. We also explore extending this work to address short-circuit defects.

VIII. Acknowledgement

This research was supported by a grant from NSF Award number CCF-2331002.

References

[1] Hapke, F., Redemund, W., Glowatz, A., Rajski, J., Reese, M., Hustava, M., ... & Fast, A. (2014). Cell-aware test. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 33(9), 1396-1409.

[2] Hapke, F., & Schloeffel, J. (2012, May). Introduction to the defect-oriented cell-aware test methodology for significant reduction of DPPM rates. In 2012 17th IEEE European Test Symposium (ETS) (pp. 1-6). IEEE.

[3] Pandey, S., Gupta, S., Natarajan, S., Sinha, A., & Chatterjee, A. (2019, November). Characterization of library cells for open-circuit defect exposure: A systematic methodology. In 2019 IEEE International Test Conference (ITC) (pp. 1-10). IEEE.

[4] Cherkauer, B. S., & Friedman, E. G. (1994). Channel width tapering of serially connected MOSFET's with emphasis on power dissipation. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2(1), 100-114.

[5] Gao, Z., Hu, M. C., Malagi, S., Swenton, J., Huisken, J., Goossens, K., & Marinissen, E. J. (2021). Reducing library characterization time for the cell-aware test while maintaining test quality. Journal of Electronic Testing, 37(2), 161-189.

[6] Franco, P., & McCluskey, E. J. (1994, April). Three-pattern tests for delay faults. In Proceedings of IEEE VLSI Test Symposium (pp. 452-456). IEEE.

[7] Singh, A. D. (2015, March). Scan based two-pattern tests: Should they target opens instead of TDFs?. In 2015 16th Latin-American Test Symposium (LATS) (pp. 1-2). IEEE.

[8] Patel, J. H. (1998, October). Stuck-at fault: a fault model for the next millennium. In Proceedings International Test Conference 1998 (IEEE Cat. No. 98CH36270) (p. 1166). IEEE.

[9] Majhi, A. K., & Agrawal, V. D. (1998, January). Delay fault models and coverage. In Proceedings Eleventh International Conference on VLSI Design (pp. 364-369). IEEE.

[10] Banerjee, P., & Abraham, J. A. (1984). Characterization and testing of physical failures in MOS logic circuits. IEEE Design & Test of Computers, 1(3), 76-86.

[11] Pomeranz, I. (2011). Multi-pattern n-detection stuck-at test sets for delay defect coverage. IEEE transactions on very large scale integration (VLSI) systems, 20(6), 1156-1160.

[12] Geuzebroek, J., Marinissen, E. J., Majhi, A., Glowatz, A., & Hapke, F. (2007, October). Embedded multi-detect ATPG and its effect on the detection of unmodeled defects. In 2007 IEEE International Test Conference (pp. 1-10). IEEE.

[13] Cho, K. Y., Mitra, S., & McCluskey, E. J. (2005, November). Gate exhaustive testing. In IEEE International Conference on Test, 2005. (pp. 7-pp). IEEE.

[14] Jas, A., Natarajan, S., & Patil, S. (2007, October). The region-exhaustive fault model. In 16th Asian Test Symposium (ATS 2007) (pp. 13-18). IEEE.

[15] Sinha, A., Pandey, S., Singhal, A., Sanyal, A., & Schmaltz, A. (2017, October). DFM-aware fault model and ATPG for intra-cell and inter-cell defects. In 2017 IEEE International Test Conference (ITC) (pp. 1-10). IEEE. .

[16] Hu, M. C., Gao, Z., Malagi, S., Swenton, J., Huisken, J., Goossens, K., ... & Marinissen, E. J. (2020, May). Tightening the mesh size of the cell-aware ATPG net for catching all detectable weakest faults. In 2020 IEEE European Test Symposium (ETS) (pp. 1-6). IEEE.

[17] Singh, A. D. (2016, May). Cell aware and stuck-open tests. In 2016 21th IEEE European Test Symposium (ETS) (pp. 1-6). IEEE.

[18] Acero, C., Feltham, D., Hapke, F., Moghaddam, E., Mukherjee, N., Neerkundar, V., ... & Zawada, J. (2015, October). Embedded deterministic test points for compact cell-aware tests. In 2015 IEEE International Test Conference (ITC) (pp. 1-8). IEEE.

[19] Liu, H. W., Lin, B. Y., & Wu, C. W. (2016, November). Layout-oriented defect set reduction for fast circuit simulation in cell-aware test. In 2016 IEEE 25th Asian Test Symposium (ATS) (pp. 156-160). IEEE.

[20] Chuang, P. Y., Wu, C. W., & Chen, H. H. (2017, September). Cell-aware test generation time reduction by using switch-level ATPG. In 2017 International Test Conference in Asia (ITC-Asia) (pp. 27-32). IEEE.

[21] Abramovici, M., Breuer, M. A., & Friedman, A. D. (1990). Digital systems testing and testable design (Vol. 2, pp. 203-208). New York: Computer science press.

[22] Lorenzelli, F., Gao, Z., Swenton, J., Malagi, S., & Marinissen, E. J. (2021). Speeding up cell-aware library characterization by preceding simulation with structural analysis. In 2021 IEEE European Test Symposium (ETS) (pp. 1-6). IEEE.

[23] Jain, A., Pradeep, W., & Glowatz, A. (2024). Optimized Timing Aware ATPG for At-Speed Test of Cell Internal Faults. In 2024 IEEE 8th International Test Conference India (ITC India), Bangalore, India (pp. 1-6). IEEE.

[24] Elmore, W. C. (1948). The transient response of damped linear networks with particular regard to wideband amplifiers. Journal of applied physics, 19(1), 55-63.

[25] Ding, L., & Mazumder, P. (2001). On optimal tapering of FET chains in high-speed CMOS circuits. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 48(12), 1099-1109.

[26] Shoji, M. (1984). U.S. Patent No. 4,430,583. Washington, DC: U.S. Patent and Trademark Office.

[27] Pandey, S., Liao, Z., Nandi, S., Natarajan, S., Sinha, A., Singh, A., & Chatterjee, A. (2021, April). Two pattern timing tests capturing defect-

induced multi-gate delay impact of shorts. In 2021 IEEE 39th VLSI Test Symposium (VTS) (pp. 1-7). IEEE.
[28] “NanGate FreePDK45 Generic Open Cell Library Si2.” [Online]. Available: https://projects.si2.org/openeda.si2.org/projects/nangatelib