Journal of Electronic Testing (2021) 37:161–189
https://doi.org/10.1007/s10836-021-05943-3

Reducing Library Characterization Time for Cell-aware Test while Maintaining Test Quality

Zhan Gao[1,2,3] · **Min-Chun Hu**[1,2,4] · **Santosh Malagi**[2] · **Joe Swenton**[2] · **Jos Huisken**[3] · **Kees Goossens**[3] · **Erik Jan Marinissen**[1,3]

Received: 16 October 2019 / Accepted: 29 March 2021 / Published online: 26 May 2021

Abstract
Cell-aware test (CAT) explicitly targets faults caused by defects inside library cells to improve test quality, compared with conventional automatic test pattern generation (ATPG) approaches, which target faults only at the boundaries of library cells. The CAT methodology consists of two stages. Stage 1, based on dedicated analog simulation, library characterization per cell identifies which cell-level test pattern detects which cell-internal defect; this detection information is encoded in a *defect detection matrix* (DDM). In Stage 2, with the DDMs as inputs, cell-aware ATPG generates chip-level test patterns per circuit design that is build up of interconnected instances of library cells. This paper focuses on Stage 1, library characterization, as both test quality and cost are determined by the set of cell-internal defects identified and simulated in the CAT tool flow. With the aim to achieve the best test quality, we first propose an approach to identify a comprehensive set, referred to as *full set*, of potential open- and short-defect locations based on cell layout. However, the full set of defects can be large even for a single cell, making the time cost of the defect simulation in Stage 1 unaffordable. Subsequently, to reduce the simulation time, we collapse the full set to a compact set of defects which serves as input of the defect simulation. The full set is stored for the diagnosis and failure analysis. With inspecting the simulation results, we propose a method to verify the test quality based on the compact set of defects and, if necessary, to compensate the test quality to the same level as that based on the full set of defects. For 351 combinational library cells in Cadence's GPDK045 45nm library, we simulate only 5.4% defects from the full set to achieve the same test quality based on the full set of defects. In total, the simulation time, via linear extrapolation per cell, would be reduced by 96.4% compared with the time based on the full set of defects.

Keywords Cell-aware test · Manufacturing defects · Open defect · Short defect · Parasitic extraction · Defect location · Equivalence · Verification · Test quality compensation

1 Introduction

Integrated circuits (ICs) today have tiny feature sizes, complex transistor architectures, and a large number of interconnect layers. Due to their large number of high-precision

Responsible Editor: L. M. Bolzani Pöhls.

✉ Zhan Gao
zhan.gao.ext@imec.be; zgao@cadence.com; z.gao@tue.nl

Min-Chun Hu
min-chun.hu.ext@imec.be; minchun@cadence.com; minchunhu@gapp.nthu.edu.tw

Santosh Malagi
malagi@cadence.com

Joe Swenton
swenton@cadence.com

Jos Huisken
j.a.huisken@tue.nl

Kees Goossens
k.g.w.goossens@tue.nl

Erik Jan Marinissen
erik.jan.marinissen@imec.be; e.j.marinissen@tue.nl

1 IMEC, Kapeldreef 75, Leuven 3001, Belgium

2 Cadence Design Systems, 1701 North Street, NY 13760 Endicott, US

3 TU Eindhoven, Den Dolech 2, Eindhoven 5612 AZ, Netherlands

4 National Tsing-Hua Univ., 101, Section2, Kuang-Fu Road, Hsinchu 30013, Taiwan

Cell: AND2X1																											
One-Cycle Detection																											
Pattern	Short Defects														Open Defects												
AB/Y	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25	d26	d27
p0=00/L	1					1	1		1		1																
p1=01/L	1	1				1	1	1			1		1														
p2=10/L				1	1	1	1		1	1	1	1															
p3=11/H	1		1	1		1			1																		
Two-Cycle Detection																											
Pattern	Short Defects														Open Defects												
AB;AB/Y	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25	d26	d27
p4=01;11/H	1		1	1		1			1						1	1	1		1		1			1			
p5=10;11/H	1		1	1		1			1						1	1	1	1	1		1						1
p6=11;01/L	1	1				1	1	1			1		1							1	1	1	1	1	1		
p7=11;10/L				1	1	1	1		1	1	1	1								1	1	1				1	1

Fig. 1 Example defect detection matrix for an AND2X1 cell

manufacturing steps, ICs are highly susceptible to manufacturing defects, and thus they need to undergo testing to weed out the defective parts before they are shipped to customers. Unfortunately, in practice, tests are not perfect either; not all defective parts are recognized during testing, which causes *test escapes*. Complex ICs are used in safety-critical applications, such as automotive and healthcare, where defects have profound consequences, and test escapes cannot be tolerated. It has always been and will continue to be a key responsibility of IC test engineers to reduce the fraction of test escapes [6].

Digital logic ICs are designed on the basis of a library of standard cells. Conventional automatic test pattern generation (ATPG) tools target faults on the boundaries of cells, such as *stuck-at* [7] and *transition* faults [27]. Intra-cell defects are typically not in the scope of conventional ATPG tools, and hence only detected fortuitously [12, 25, 28]. Cell-internal defects cause a significant fraction of test escapes [6]. Cell-aware test (CAT) reduces test escapes by explicitly targeting cell-internal defects [18]. The general CAT methodology consists of two stages. In Stage 1, *library characterization*, we first determine, for each library cell based on its layout, the set of locations where cell-internal open and short defects might occur, and model defects using selected resistors. Subsequently, defect characterization utilizes dedicated analog simulation to determine which cell-level test pattern detects which cell-internal defect; the result is encoded in a *defect detection matrix* (DDM) per cell. An example DDM for an AND2X1 cell is shown in Fig. 1. A DDM is a binary matrix. Its rows correspond to cell patterns, which contain stimulus bits for all cell inputs and the corresponding defect-free response bit on a single cell output. The columns correspond to defects that are identified to occur in the cell potentially. A DDM entry denotes whether a cell pattern detects a defect (denoted by ‘1’) or not (empty denotation). The ‘1’s in each pattern row show which defects are detected by this cell pattern; the ‘1’s in each column indicate the cell patterns detecting the corresponding defect. Stage 2, *cell-aware ATPG*, uses as inputs the IC netlist with library cells as its building blocks and the set of DDMs for these library cells. For each cell instance in a chip design, an intra-cell defect is covered if there is at least one cell pattern (1) which, according to the DDM, detects this particular defect, and (2) which the cell-aware ATPG tool is able to successfully expand from cell-to-chip level. After a cell pattern is expanded successfully, fault simulation identifies all the other covered defects in the pattern expansion path.

This paper focuses on Stage 1, library characterization, as it is fundamental to both test quality and test cost of CAT. The set of potential defects should be realistic and complete. Too few defect locations cause the test to miss defects; this negatively impacts the test and product quality. Too many defect locations imply unnecessary time-consuming analog simulations during defect characterization. We propose a library characterization tool flow that covers all cell-internal defects to guarantee the test quality and significantly reduces the defect simulation time with maintaining the test quality.

We automatically identify a *full set* of potential locations of open defects on and short defects between both intra-cell interconnects and transistor terminals. During the full set of defect-location identification (DLI), we use parasitic extraction (PEX) not only in its conventional role to create accurate analog simulation models of library cell layouts, but also to analyze the cell layouts to identify possible defect locations on and between interconnects. The number of identified defect locations quickly grows large, even for small library cells, making the time cost of the following defect simulation unaffordable. However, many defects in the full set are equivalent with respect to their logic fault behaviors, and hence require the same test patterns, namely cause identical DDM columns. To reduce the number of defects to be simulated while maintaining the full-set-based test quality, we first collapse the full set to a subset compact set by grouping equivalent defects and selecting only one defect per group into the compact set, which is used as input of the defect simulation. With the assistance of defect simulation, we verify if all collapsed defects cause identical DDM columns. An *error* is determined if a collapsed group of equivalent defects contains at least one defect non-equivalent to the others. A non-equivalent defect in a collapsed defect group causing one DDM column to be missed for this group negatively impacts the test quality. Some errors are specific to simulation user settings, and hence cannot be avoided before the simulation. Subsequently, we eliminate errors by identifying and adding the missed DDM columns with additional simulations, thus compensate the test quality to the same as that based on the full set of defects. The experiment results show that for 351 combinational cells in Cadence's GPDK045 45nm technology, we reduce 96.4% simulation time comparing with simulating the full set of defects. This paper extends our work in [11] with a refined PEX cell model, an improved solution for reducing the defect simulation time with maintaining the test quality, and additional experiment results on defect simulation.

The remainder of this paper is organized as follows. Section 2 reviews related prior work. The proposed library characterization tool flow is introduced in Section 3. Section 4 describes a cell model used for the PEX and the general PEX settings for the defect-location identification. To reduce the defect simulation time and maintain the test quality, we describe how we handle the open and short defects in Section 5 and 6, respectively. Section 7 presents experiment results and Section 8 concludes this paper.

2 Related Prior Work

CAT identifies only those cell patterns that actually contribute to the detection of intra-cell defects; this requires knowledge of which defects to expect. For this, CAT is inspired by layout-level defect-to-fault analysis methods like inductive fault analysis (IFA) [32], which are intractable to be applied at chip level, but very feasible if applied on individual library cells. CAT was introduced by Hapke et al. in 2009 [15]. Initially, only hard (low ohmic) short defects (1Ω) in combinational logic cells were targeted. Later, this was extended to hard (high ohmic) open defects (1GΩ), weaker short defects (1Ω – 20kΩ) [13, 16], and sequential cells such as scan flip-flops [14]. Several studies have shown that CAT offers superior test quality and reduces test cost in comparison with *n*-detect [18, 25, 28], embedded multi-detect [12, 38], or gate-exhaustive test [5, 17]. Industrial application has provided experimental evidence of the effectiveness of CAT to reduce test escapes [13, 18, 19, 33, 37]. CAT also improves the diagnosis accuracy and efficiency by recording the layout location of potential defects [26, 36].

The quality improvement promised by CAT critically depends on the details of the DLI and defect characterization steps, as they determine which intra-cell defects will be considered and which cell-patterns can actually contribute to the detection of these defects. No prior work discloses which intra-cell defects exactly are modeled and when the defect set is pruned, how that is done, and what the effect is on the compute time for analog defect simulation. Other papers use PEX for DLI, but do not report how the many user controls of a PEX tool were set [14, 15, 34]; in this paper, we fully specify all these details. Prior publications present different opinions regarding the inclusion of inter-layer shorts: Hapke et al. include them [14, 15], while Liu et al. claim they do not occur and hence can be omitted [24]. We support both options, as we have observed that in advanced-node technologies, inter-layer shorts can indeed occur [10], but do not want to burden a user of more mature technology nodes with intra-cell defects that are indeed very unlikely to occur in these nodes. In addition to intra-cell defects, [14] includes also shorts, opens, stuck-at, and transition faults at the cell I/Os in cell-aware ATPG. In this paper, we demonstrate that these cell-boundary defects are often equivalent to already modeled cell-internal defects. In those cases, our approach does not require to explicitly include them into the defect set. In this paper, we also assess the defect equivalence from the analog simulation perspective to reduce the number of defects to be simulated. To maintain the test quality, which is the most important test metric, we use the simulation to efficiently verify the defect equivalence.

3 Library Characterization Flow

The proposed CAT library characterization invoking several other EDA tools along the way is depicted in Fig. 2. Step 1, for each cell in a library, PEX with dedicated user settings is performed. The output is the cell's transistor-level netlist with extracted parasitic resistors and capacitors. Step 2, on the basis of the cell's netlist, we determine a full set of defect locations with possible open- and short-defect locations. This approach is embedded in Cadence's tool Modus. Parasitic resistors on and capacitors between cell-internal interconnects are all locations of open- respectively short-defect candidates for the interconnects. Transistor-internal defects are modeled by opens on and shorts between transistor terminals. However, the

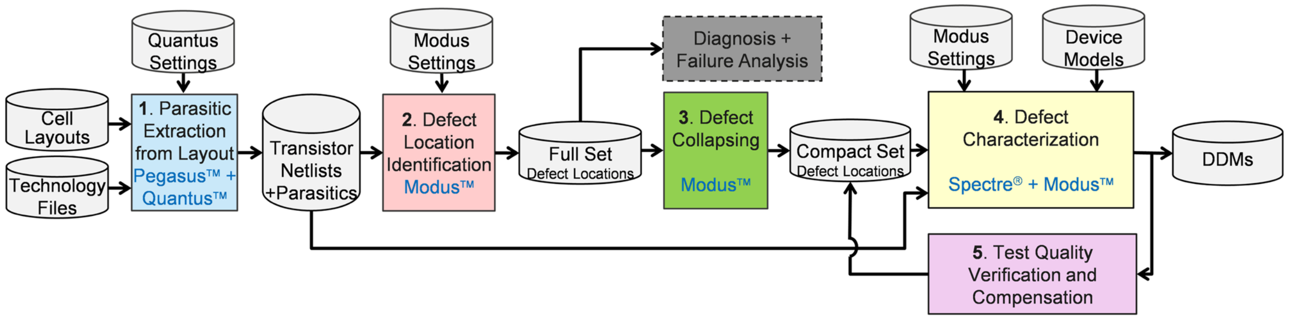


Fig. 2 Library characterization consists of five steps: (1) parasitic extraction, (2) defect-location identification, (3) defect collapsing, (4) defect characterization, and (5) test quality verification and compensation

full set of defect locations can be large just for a single cell. Simulating the full set of defects takes a lot of time, which is impractical. For example, for the AND2X1 cell in Cadence's GPDK045 library, the full set contains 61 open defect locations and 769 short defects.

To save downstream defect characterization time, In Step 3, we *collapse* the full set to a compact set based on the defect equivalence [11]. As equivalent defects are detected by the same set of cell patterns which is identified by analog simulation, per group of equivalent defects simulating only one defect is enough. Also, during the cell-aware ATPG, any one of the required cell patterns is expanded successfully, all of the equivalent defects are detected simultaneously. The advantage of the defect collapsing algorithm is that computing time is negligible compared with the simulation time and genetic to all technologies. However, due to the effect of specific simulation user settings and intra-cell parasitic resistors and capacitors dependent on technology, it is possible that not all the collapsed defects are equivalent, and hence we would miss some defects, which are reflected as missing DDM columns. To maintain the test quality based on the full set of defects, we add the missed DDM columns by additional simulations in Step 5.

In Step 4, Modus' defect characterization function adds a resistive value to each defect location in the compact set; this allows us to model hard as well as weak resistive open and short defects [20]. The defects and the defect-free netlist are submitted to the analog simulator. Each defect is injected into the defect-free netlist. For each short defect, we simulate the defective netlist for the exhaustive set of one-cycle cell patterns to detect a static fault; for each open defect, we simulate the defective netlist a set of two-cycle cell patterns to detect a delay fault. Only if the simulation for the combination of defect d and cell pattern p gives a response which differs from the defect-free cell on at least one output of the cell, we mark in the DDM entry for defect d and pattern p as 'detected'. For *all* library cells, *all* defects are simulated with *all* possible cell patterns, which generates per library cell a DDM as the output of the defect characterization.

In Step 5, with the assistance of the upstream defect characterization, we verify if all defects in each collapsed group of defects result in identical DDM columns. If not, we identify missed DDM columns to compensate the test quality.

Defect simulation is time consuming, due to the fact that it contains three nested loops: for all cells, for all defects, and for all cell patterns. Fortunately, this task needs to be executed only once per library release, while the resulting DDMs can be reused for all IC designs based on the same library. Some defects cannot be detected by any cell pattern. These non-detectable defects do not affect cell functionality even if they are present. Therefore, lower defect coverage is a good sign which indicates the cell is not easy to be affected by the potential defects, namely the cell design is robust. Only the detectable defects continue to Stage 2, cell-aware ATPG, as cell-internal faults.

4 PEX and Its Cell Model

The conventional role of parasitic extraction (PEX) in IC design is to determine the non-ideal electrical behavior of on-chip interconnects. This behavior is typically not part of the original design intent (and hence the term 'parasitic'), but is present nevertheless and therefore needs to be assessed in order to build an electrically accurate simulation model of the circuit in question. The electrical non-idealities in the circuit's interconnects are lumped into resistors R, capacitors C, and inductors L, which are added into the circuit's original netlist [29]. In standard cells, interconnect lengths are very small, and hence the resulting L values are very small and therefore can be ignored [21].

A *net* that electrically connects two or more *terminals* is divided into net *segments*, which are bounded by terminals and/or *internal nodes*. Below, we present a mathematical description of the cell model as generated by the PEX tool. Let *Cells* be the set of library cells. Let $Terminals_c$ be the set of net terminals of library cell $c \in Cells$. $Terminals_c$ contains the inputs and outputs of cell c (a.k.a. *ports*), power (V_{DD})

and ground (V_{SS}), and the source, drain, gate, and bulk terminals of the transistors in c. Let $IntNodes_c$ be the set of all internal net nodes of library cell c; internal nodes are the points in a net where the PEX tool starts a new net segment. With $Nodes_c$ we denote the set of all nodes, i.e., $Nodes_c = Terminals_c \cup IntNodes_c$.

The extracted parasitic resistors and capacitors can be represented by two weighted undirected graphs, $G_R = (V, E_R)$ and $G_C = (V, E_C)$, with a common set of vertices $V = Nodes_c$, but different edge functions $E_R = R(i,j)$ and $E_C = C(i,j)$, with R, C: $NodesR, C : Nodes_c \times Nodes_c \longrightarrow \mathbb{R}^+$ where $R(i, j)$ and $C(i, j)$ specify respectively the parasitic resistance and capacitance between nodes i and j as extracted by the PEX tool.

We define the abbreviation function *segm*: $Nodes_c \times Nodes_c \longrightarrow Bool$[1] to denote that nodes i and j are electrically connected by a net segment:

$$\mathrm{segm}(i,j) \equiv \mathrm{segm}(j,i) \equiv R(i,j) \tag{1}$$

The function *conn_net*: $Nodes_c \times Nodes_c \longrightarrow Bool$ is the transitive closure of *segm*, denotes that two nodes are electrically connected through zero or more net segments, and is defined as follows:

$$\mathrm{conn\_net}(i,j) \equiv \&(i = j) \vee \mathrm{segm}(i,j) \vee \exists_{k \in \mathrm{Nodes}_c}(\mathrm{conn\_net}(i,k) \wedge (\mathrm{conn\_net}(k,j)) \tag{2}$$

The function *net*: $Nodes_c \longrightarrow \mathcal{P}(Nodes_c)$ yields, for a given node i, the *net* of i, i.e., the set of nodes with which i is electrically connected in cell c:

$$\mathrm{net}(i) = \{j \in \mathrm{Nodes}_c \mid \mathrm{conn\_net}(i,j)\} \tag{3}$$

The net function effectively partitions the set $Nodes_c$ in a number of disjoint non-singleton nets, such that the following holds:

$$\forall_{i \in Nodes_c}(i \in net(i) \wedge |net(i)| \geq 2) \wedge \tag{4}$$

$$\forall_{i,j \in Nodes_c}(j \in net(i) \Rightarrow i \in net(j)) \tag{5}$$

The set of disjoint nets in cell c is $Nets_c = \bigcup_{i \in Nodes_c}\{\{net(i)\}\}$. Each node $n \in Nodes_c$ resides in a processing layer. Let *Layers* be the set of processing layers. Layer function *lyr*: $Nodes_c \longrightarrow Layers$ gives the layer of a node. For $i, j \in Nodes_c$ with $segm(i, j)$, either (1) $lyr(i) = lyr(j)$ and the net segment between i and j is said to be in $lyr(i)$, or (2) $lyr(i) \neq lyr(j)$ and the net segment between i and j is a vertical interconnection between different layers, called a *via* or *contact*. The PEX tool can list the x, y layout coordinates of each node.

[1] *Bool* denotes the set with Boolean values *true* and *false*.

A net can fork to multiple destinations. We define *Forks*, the set of *fork nodes* in cell c, as follows:

$$\begin{aligned} Forks_c = &\{i \in Terminals_c \mid |\{j \in net(i) | segm(i,j)\}| > 1\} \cup \\ &\{i \in IntNodes_c \mid |\{j \in net(i) | segm(i,j)\}| > 2\} \end{aligned} \tag{6}$$

The function *conn_branch*: $Nodes_c \times Nodes_c \longrightarrow Bool$ denotes that two nodes are part of the same net branch, and is defined as follows:

$$conn\_branch(i,j) \equiv ((i = j) \vee segm(i,j) \vee \exists_{k \in Nodes_c \setminus Forks_c}(conn\_branch(i,k) \wedge conn\_branch(k,j))) \tag{7}$$

The function *branch*: $Nodes_c \longrightarrow \mathcal{P}(Nodes_c)$ yields, for a given node i, *the branch of* i, i.e., the set of nodes that are part of the same net branch as i:

$$\mathrm{branch}(i) = \{j \in Nodes_c | conn\_branch(i,j)\} \tag{8}$$

The set of all branches in cell c is $Branches_c = \bigcup_{i \in Nodes_c}\{\{branch(i)\}\}$.

We illustrate the model described above with an inverter cell INVX1 from the Cadence GPDK045 cell library [3] as example in Fig. 3. Figure 3a shows the layout. We marked out in the schematic (see Fig. 3c) the in total twelve terminals: one input port (A), one output port (Y), two transistors with four terminals each, V_{DD}, and V_{SS}. Figure 3a shows in the PEX model each terminal exists in one of four layers: n-diffusion, p-diffusion, poly, and metal-1. V_{DD}/V_{SS} and cell input/output ports are on the metal-1 layer. In addition, the PEX tool has identified eight internal nodes. The total set of 20 nodes partitions into four disjoint nets: $Nets_{\mathrm{INVX1}} = \{\{A, A\#1, A\#2, G1, G2\}, \{Y, Y\#1, Y\#2, D1, D2\}, \{V_{DD}, V_{DD}\#1, V_{DD}\#2, S1, B1\}, \{V_{SS}, V_{SS}\#1, V_{SS}\#2, S2, B2\}\}$. Contacts exist from metal-1 to the other three layers. Net $\{A, A\#1, A\#2, G1, G2\}$ forks out to two destinations, gate terminals $G1$ and $G2$, and contains three branches $\{\{A, A\#1, A\#2\}, \{A\#2, G1\}, \{A\#2, G2\}\}$. Similarly, the other nets also can be broken up into branches.

In cells more complex than INVX1, two adjacent transistors in the layout are designed to share the same diffusion area as their source or drain. However, in the DSPF format netlist, the PEX tool assigns a particular name for each transistor terminal even though two different terminal names represent the same diffusion area. Consequently, the interconnection between such two terminals is presented as a *virtual* 0.001Ω net. Even though the 0.001Ω resistor does not physically exist, its resistance is too small to impact the simulation results. The real parasitic resistance of the shared diffusion area is extracted and packaged into the device model that is also included in the extracted netlist for an accurate simulation. Figure 4 shows an example virtual interconnect in the AND2X1 cell, designed as a NAND circuit followed by an inverter, from Cadence's GPDK045 library [3]. The two NMOS transistors in the NAND part share the same

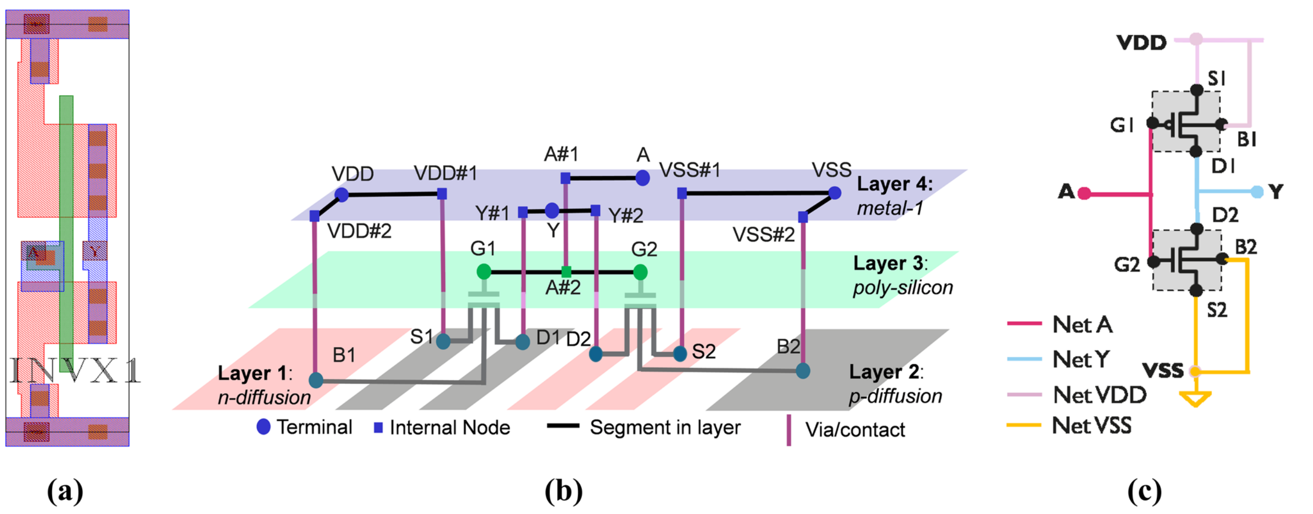


Fig. 3 Example library cell INVX1: **(a)** layout, **(b)** schematic, and **(c)** PEX cell model

diffusion area, and hence no physical interconnect between the two NMOS transistors. The interconnection between the terminals *S* and *D* shown in the schematic is extracted as a virtual net.

We use Cadence Quantus v18.1 to perform PEX. Figure 5 lists the general PEX settings. The remaining PEX settings specific to short- and open-defect location identification are described in the following two sections. Users should set the input database format generated by a layout versus schematic EDA tool. We use the '*pvs*' database generated by Cadence Pegasus (Line 1). We need to extract resistor (*R*) and capacitor (*C*) values (Line 2). We use DSPF (Cadence's detailed standard parasitic format) as output format (Line 3). The benefits in the CAT context of DSPF over, for example, the well-known Spice format [35] are that DSPF format [2] features (1) an explicit layer list, (2) explicit listing of nets, and (3) explicit listing of net segments in diffusion layers. For usage during diagnosis, we include as comments into the netlist for the parasitic *R*s and *C*s: layer information, segment widths, and *x*, *y* location coordinates (Lines 4–6).

5 Open-Defect Identification and Characterization

This section describes how we handle open defects in the library characterization. In section 5.1, we guide the PEX tool to extract parasitic resistors at all potential open-defect locations on interconnects. Afterward, we identify the full set of open-defect locations for both interconnects and transistor terminals in Section 5.2. Section 5.3 analyzes the electrical behaviors of open defects with the aim to identify their equivalence, which is the basis of open-defect collapsing and test quality verification. The algorithms of open-defect collapsing and characterization are presented in Sections 5.4 and 5.5, respectively. Section 5.6 utilizes the simulation results for the compact-set open defects to verify if the test quality is maintained the same as that based on the full set. If not, we compensate the test quality.

Fig. 4 An example virtual net in the GPDK045 library cell AND2X1

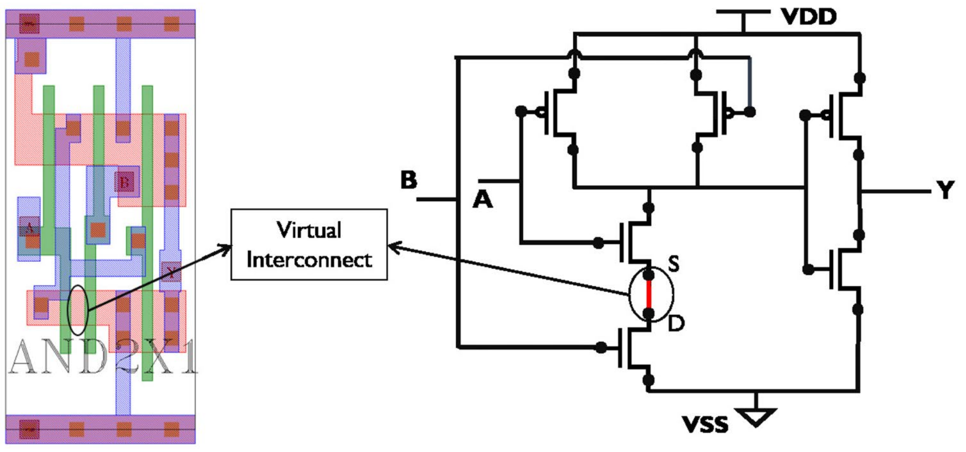


Fig. 5 General Cadence's Quantus settings for cell-aware DLI

Quantus Settings 1 [General PEX Settings]

```
1: input_db -type pvs;
2: extract -type rc_coupled;
3: output_db -type dspf;
4: output_db -include_parasitic_res_model comment;
5: output_db -include_parasitic_res_width true;
6: output_db -output_xy PARASITIC_RES PARASITIC_CAP;
```

5.1 PEX for Open-defect Location Identification

An open defect on an interconnect might occur between any two connected nodes (i, j) with $segm(i, j)$. We analyze for each library cell its layout by letting the PEX tool extract all net segments and their parasitic resistor values. Each extracted segment $segm(i, j)$ with its parasitic resistor $R(i, j)$ is considered as a potential open-defect location on the interconnects. Reasons for us to let a PEX tool split a net into multiple segments are (1) a *fork* in a net with multiple destinations, as open defects on different *branches* of the fork affect different terminals, hence should be separated; (2) a vertical interconnect from one layer to another (referred to as a *via* which in library cells is typically called "contact"); or (3) a 90° bend ('L shape') within a layer, as the contacts and the 90° bends are sensitive locations of manufacturing defects.

Figure 6 lists the settings of the user options of Quantus to identify the full set of open-defect locations on intra-cell interconnects. In Line 1, we instruct Quantus *not* to break up segments into multiple segments only because the layout length of the segment exceeds a limit. The two options in Lines 2–3 specify that defect-sensitive circuit elements are included in the PEX output netlist: vias, and 90° bends in nets – the latter is identified by electromagnetic analysis.

Parallel segments (other than parallel vias and contacts) are expected not to occur in the heavily-optimized circuits that standard-cell libraries are (Line 4). The only type of parallel R segments that occur frequently in library cells are parallel vias. The inherent redundancy of multiple parallel vias is meant to increase the yield, and effectively work as one (lower-resistance) parallel connection. Lines 5–6 merge them into one segment with replacement R value. To control complexity, our CAT flow works on the basis of one defect insertion at a time. However, by varying the resistance value of the open defect in the merged via, we can model any number of parallel vias as being defective. We suppress listing of parasitic resistors in dangling segments (Line 7). As the missing R's do not affect the simulation results and open defects in dangling segments would not cause faulty behavior, we utilize this option. This option does not affect the extraction of capacitors to the dangling segment nodes, and therefore it does not affect the fault simulation results.

A R value is indicative of the probability that the corresponding segment will suffer from an open defect: a large R value indicates that the corresponding net segment is thin and/or long and both increase the probability for that interconnect to be affected by an open defect. The parameter min_res in Line 8 filters out parasitic resistors below a specified limit (in Ohm) during PEX. We want the PEX tool to include *all* resistor segments in the cell model as open-defect locations, and set a user-defined option R in the DLI function to filter out some open-defect locations with small Rs based on users' requirements. Quantus accepts a min_res value only larger than 0, therefore we specify min_res low enough (i.e. $10^{-10}\,\Omega$) to guarantee that all segment Rs are extracted.

5.2 Full Set of Open-defect Locations

A standard cell is built up from interconnected transistors. We consider open and short defects for both interconnects and transistors. With the dedicated user settings of the PEX tool, parasitic resistors indicate the potential open-defect locations on cell-internal interconnects. For transistors, we adapt transistor-internal defects to transistor-boundary defects. For example, bad doping causes high impedance at source or drain, which can be modeled by adding

Fig. 6 Dedicated Quantus settings for open-defect location identification

Quantus Settings 2 [for Open-Defect Location Identification]

```
1: extraction_setup -max_fracture_length infinite;
2: output_db -add_explicit_vias true;
3: output_db -em_extract true;
4: filter_res -merge_parallel_res true;
5: filter_res -merge_parallel_via true;
6: filter_res -max_via_array_size auto;
7: filter_res -remove_dangling_res true;
8: filter_res -min_res 1e-10;
```

high-ohmic resistors at source or drain terminal [23]; corrosion of the gate metallisation or poor gate etching results in a loss of the gate controllability, which can be modeled by gate terminal open [8, 9]. By default, opens on source, drain, and gate terminals are considered for transistors.

The identification of the full set of open-defect locations is described in Algorithm 1. This algorithm has several following controls with which the user can authorize the tool to filter out certain open locations. The default settings for these open-defect locations are such that no locations are filtered out and thus best test quality can be obtained.

– Parameter *disableTrTerminalOpens* allows to explicitly disable the identification of open-defect locations on transistor terminals (default: *false*).
– *TrTerminalOpenSet* specifies which set of transistor terminals are considered as possible open locations (default: {*source, drain, gate*}).
– Threshold function $R_{\mathrm{th}}(\ell)$ defines, per layer ℓ, that extracted segment resistors will *not* be identified as open-defect locations if $R < R_{\mathrm{th}}(\ell)$ (default: $R_{\mathrm{th}}(\ell) = 0$ for all layers ℓ).

In Line 2 of Algorithm 1, the full set of open-defect locations is initialized. Open-defect locations are identified on all physical segments of which the parasitic resistor values exceed the user-defined threshold $R_{\mathrm{th}}(\ell)$ (Lines 03–09). Next, if the user has not disabled the identification of transistor open locations, for all transistors, the open-defect locations are identified on all terminals in *TrTerminalOpenSet* (Lines 10–14).

5.3 Open-Defect Equivalence

For an electrical test, open defects on different *branches* of a fork structure affect different downstream destinations, and therefore should be characterized separately. On each branch, the open defects on concatenated segments block logic value transition for the same destination, namely the same transistor(s), such that can cause equivalent fault effects. Therefore we consider to collapse the open defects per branch into one open. However, due to the impact of parasitic capacitors, on some branches, the open defects per branch lead to slightly different electrical effects. To maintain the test quality, we should make sure the variation on the open-defect electrical behavior on the same branch does not influence the detection results, such that the resulting DDM columns of these open defects are identical. We analyze the open-branch behaviors and filter out the branches that requires test quality verification after the simulation. In the following, we classify the branches into four types, and for each of them study the difference of electrical behaviors of segment open defects on the same branch.

Type 1: if a branch consists of only one segment, we classify it into Type 1. On this branch, at most one open defect is determined, and we cannot collapse any open defects anyway. Therefore, the test quality is always maintained.

Type 2: the branch is a part of net V_{DD} or V_{SS} and is also on power rail (i.e. in metal-1 layer). On the power rails of standard cells in a circuit design, multiple voltage sources are designed and implemented with contacts that

Algorithm 1 FULL-SET OF OPEN-DEFECT LOCATION IDENTIFICATION

```
01: for all $c \in Cells$ do {
02:   $openFullSet_c = \varnothing$;
03:   for all $i, j \in Nodes_c$ do {
04:     if $segm(i,j) \wedge$ segment $(i,j)$ non-virtual then {
05:       if $lyr(i) = lyr(j) \wedge R(i,j) \geq R_{\mathrm{th}}(lyr(i))$ then {
06:         $openFullSet_c = openFullSet_c \cup \{(i,j)\}$; }
07:       if $lyr(i) \neq lyr(j) \wedge R(i,j) \geq R_{\mathrm{th}}(lyr(i), lyr(j))$ then {
08:         $openFullSet_c = openFullSet_c \cup \{(i,j)\}$; }
09:   } }
10:   if $\neg$ $disableTrTerminalOpens$ then {
11:     for all transistors $t$ in $c$ do {
12:       for all $tm \in TrTerminalOpenSet$ do {
13:         $openFullSet_c = openFullSet_c \cup \{t.tm\}$;
14: } } } }
```

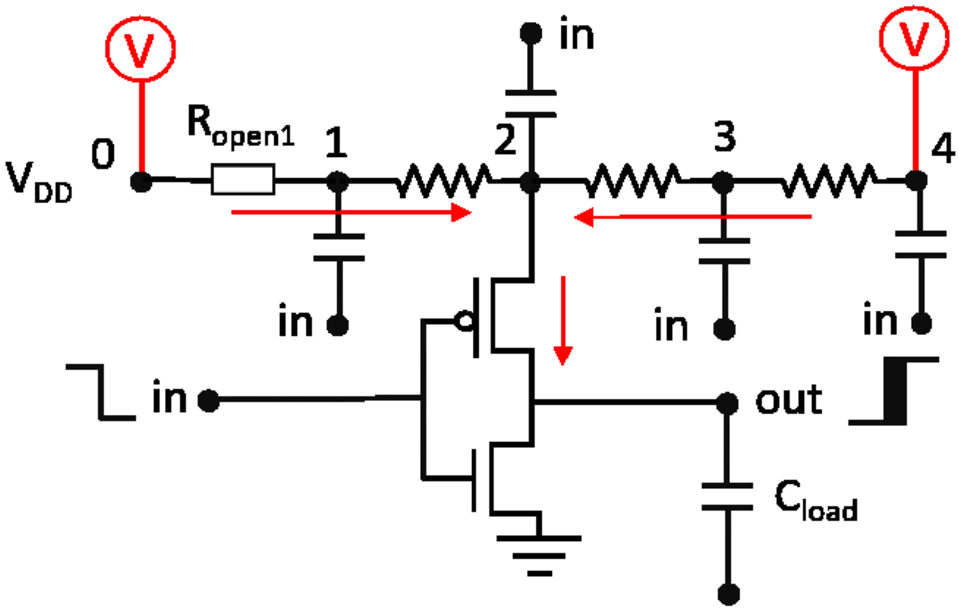


Fig. 7 Different open locations on a power net branch

are from the external power supplier to the power rails. Signal open defect cannot block the voltage source for transistors. The concept is shown in Fig. 7. Even though R_{open1} blocks the voltage source on the left side, the voltage source on the right side still serves. Therefore, the two open defects on the branch that consists of *segm*(0, 1) and *segm*(1, 2) do not have a different effect on the circuit.

Type 3: if a branch consists of multiple segments and an open defect on such a branch blocks the signal transitions for all downstream PMOS or NMPS gate terminals, we classify such a branch into Type 3. A net can drive signal or multiple parallel CMOS transistor gate terminals. An example net structure driving multiple transistors is shown in Fig. 8. The net connecting six transistor drain terminals and six transistor gate terminals is split into 19 branches of which numbers are marked beside the branches. We use different colors for adjacent branches. Among Branches 7, 10, 11, and 12 which consist of multiple segments, only Branch 10 is in Type 3. On a Type 3 branch, the open-defect location affects the caused delay size. We explain later.

Type 4: the a branch consists of multiple segments but an open defect on this branch cannot block the signal transitions for all downstream PMOS or NMPS gate terminals. This branch type can be found in standard cells with high drive strength, as cell designer use multiple parallel transistors controlled by the same net to increase the drive strength. These multiple parallel transistors are switched on simultaneously, such that large conductive current through the transistors quickly charge or discharge the downstream components. In Fig. 8, among Branches 7, 10, 11, and 12 which consist of multiple segments, Branches 11 and 12 are in Type 4.

Open defects at different locations result in different *RC* networks of a branch. Figure 9a shows an example *RC* network of a defect-free branch. The capacitive coupling

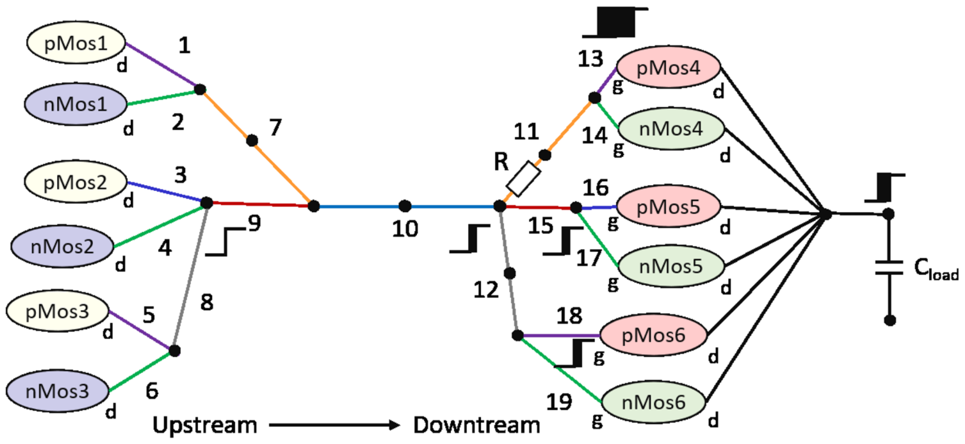


Fig. 8 An example net in a cell with high drive strength

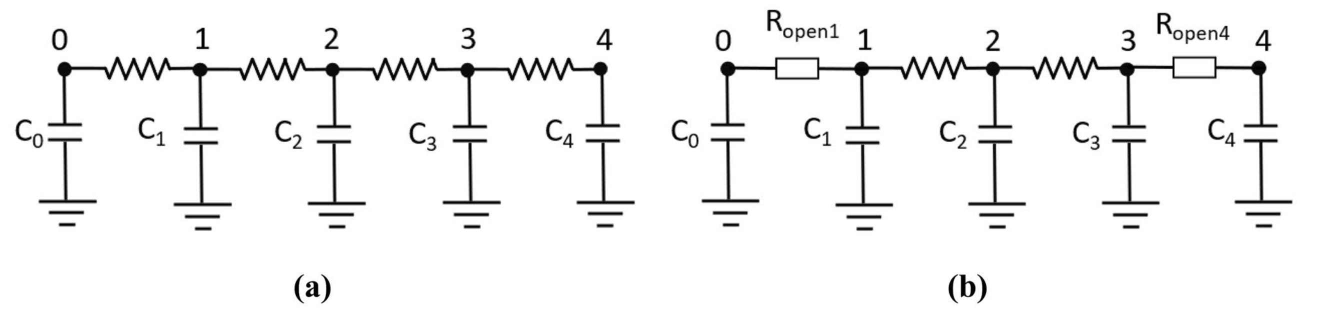


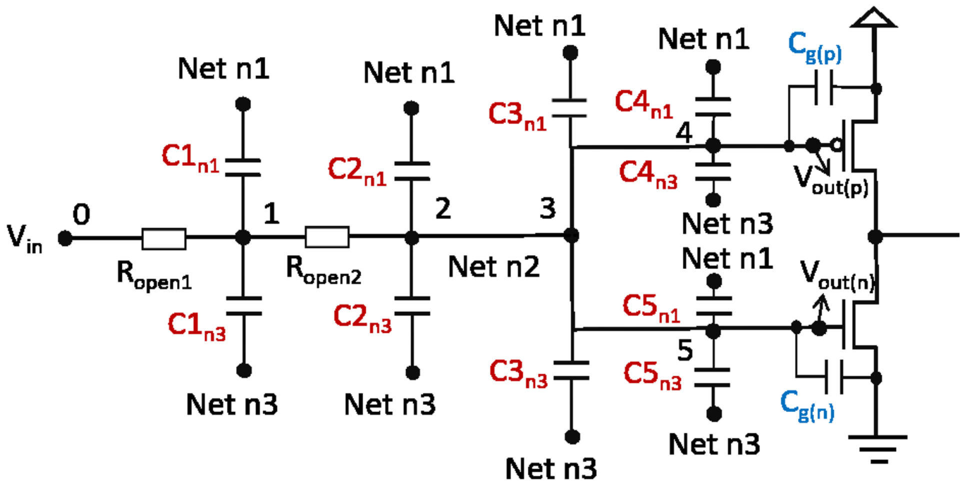
Fig. 9 (**a**) *RC* network of a defect-free branch. (**b**) *RC* network with two opens at different locations

between the branch and ground net V_{SS} is extracted by the PEX tool as capacitors distributed based on the branch nodes. Figure 9b shows two open defects at different locations on the branch. As the branch's parasitic resistance is significantly smaller than the open-defect resistance, the time constant of delay caused by either open defect is determined by the open-defect resistance and the capacitance from the open resistor to the downstream branch terminal [30]. If we assume node 4 is closer to the downstream transistor network than node 0, the time constants of the delays due to *open1* and *open4* are $R_{open1} \cdot (C_1 + C_2 + C_3 + C_4)$ and $R_{open4} \cdot (C_4)$, respectively. With $R_{open1} = R_{open4}$, the open-defect location affects the delay size through the variation of load capacitance calculated by summing the capacitance from the open-defect location up to the branch's downstream terminal.

Please note, in a standard cell the load capacitance after an open defect is the sum of capacitances from the open-defect location up to the downstream transistor gate terminals. It consists of two components: (1) interconnect capacitance and (2) the total gate capacitance of the driven transistors [30]. Figure 10 shows the load capacitances after R_{open1} on Net n2 that is driving an inverter. The two components of the load capacitance are drawn in red and blue colors, respectively. Eight nodes, internal nodes 0–5 and two gate terminals, split Net n2 into seven segments and three branches. Net n2 has two neighboring nets, n1 and n3. The parasitic capacitances between n2 and the two neighboring nets are distributed based on each node of n2. Ci_j denotes the capacitance distributed between node *i* and neighboring net *j*. Ci_j is calculated by summing the values of extracted capacitors of which one coupling node is *i* and the other coupling node is on net *j*. Two open defects *open1* and *open2* are at different locations on the same branch of net n2. During the defect characterization, we inject only one open defect each time. Assuming the resistances of *open1* and *open2* are equal, *open1* causes a longer delay than *open2* due to the extra interconnect capacitors $C1_{n1}$ and $C1_{n3}$.

The interconnect capacitance after an open defect is also affected by the status of logic values on the neighboring nets. A resistive open defect causes a delay fault and hence can be detected by two-cycle test patterns. We perform simulations of a set of two-cycle cell patterns; each of them contains only one cell input transition and makes at least one cell output transit from the first to the second cycle. The two-cycle cell patterns cause either a static or a dynamic logic value on each neighboring net. In the following, we discuss the effect of static and dynamic neighboring nets on the interconnect capacitance after an open defect.

Fig. 10 Circuit model of two opens at different locations on a branch



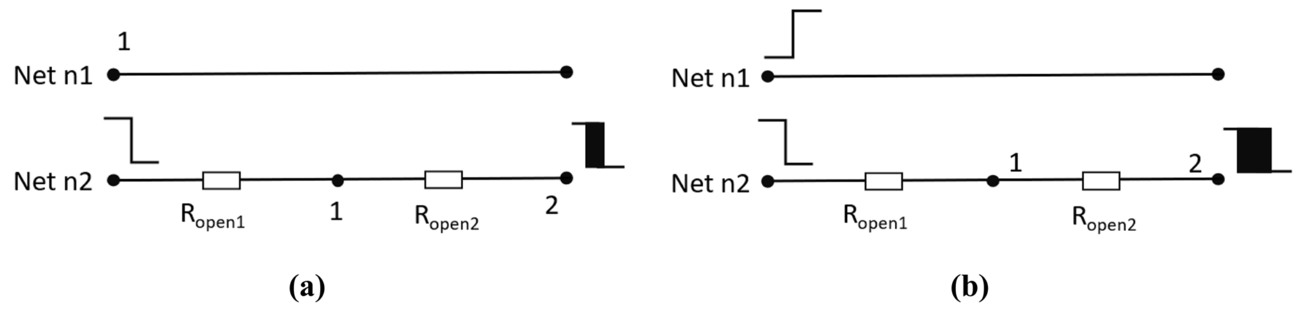


Fig. 11 **(a)** Neighboring nets with static logic values. **(b)** Neighboring nets the opposite transition to the open branch

Figure 11a shows from the first to the second cycle of a cell pattern, the logic value on the open net, n2 changes from 1 to 0, while the logic values on its neighboring net stays static. The node capacitances between the open branch and its neighboring net are equal to the extracted capacitance values. Figure 11b shows the situation that the logic value on the neighboring net is with opposite transition to n2. In the first cycle of the cell pattern, due to the voltage difference on Net n2 and Net n1, Net n2 with high voltage value charges the parasitic capacitors between Net n1. In the second cycle, the voltage value on Net n1 is high while the voltage value on Net n2 is low, the parasitic capacitors between n1 and n2 are first discharged and then charged. Therefore, the equivalent load capacitance between Nets n1 and n2 are 2× extracted capacitance. Consequently, neighboring nets with opposite transition to the open branch increase the delay size. Similarly, a neighboring net with the same transition with the open branch decreases the delay size. [1] builds an indication function $I_n(p)$ to evaluate the impact of logic value state on $|l|$ neighboring nets. $I_n(p)$ is approximated as Equation 9.

$$I_n(p) = \begin{cases} 0, \text{for the same transition in neighboring net } n \text{ for pattern } p \\ 1, \text{for the static state in neighboring net } n \text{ for pattern } p \\ 2, \text{for the opposite transition in neighboring net } n \text{ for pattern } p \end{cases} \tag{9}$$

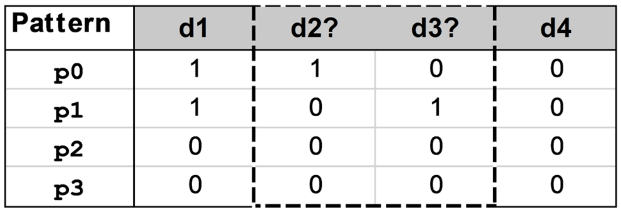
Given an open defect on a net segment $segm(i,j)$ with $|l|$ neighboring nets, the function $open\_net$: $Nodes_c \times Nodes_c \longrightarrow \mathcal{P}(Nodes_c)$ gives the set of nodes from the open-defect location to the downstream gate terminals:

$open\_net(i,j) = \{net(j)|segm(i,j) = False\}$, with node j is closer to the downstream gate terminals. Therefore, the equivalent capacitance from open-defect to the end of the open net with respect to a cell pattern is calculated by

$$C_L(p) = \sum_{k \in open\_net(i,j)} \sum_{n \in \{1..l\}} (Ck_n \cdot I_n(p)) + (\sum C_g) \tag{10}$$

The first addend is the interconnect capacitance after the open-defect location which affects the addend value through the set of nodes $open\_net(i,j)$ after the open segment.

The delay size caused by an open on a Type 3 branch determines the delay size measured at cell output, as the Type 3 branches are the only way for a signal transition propagated to the cell output. The effect of open-defect location on delay size can be significant. But on Type 4 branches, the open defect does not affect all signal propagation ways. The effect of open-defect location on delay size at cell output can be ignored. For example, in Fig. 12 the difference between open defects on Branches 10 and 11 (or 12) is that open on Branch 10 affect all downstream transistors while open on Branch 11 only effect *pMos4* and *nMos4*. Open defects on the two segments of Branch 10 cause different delay sizes due to their different load capacitances. Either open causes the signal on this net arrives at each driven gate terminal with the same delay size. However, open defects on Branch 11 cannot block the signal for transistors *pMos5*, *nMos5*, *pMos6*, and *nMos6* which still perform well. In defect-free situation, either {*pMos4*, *pMos5*, *pMos6*} or {*nMos4*, *nMos5*, *nMos6*} are conducted simultaneously; with an open defect on Branch 11, only {*pMos5*, *pMos6*} or {*pMos5*, *pMos6*} are conductive simultaneously. Even though a high-ohmic open resistor results in a serious delay at the gate terminal of *pMos4* or *nMos4*, the delay size at the output of the six transistors is determined by the performance difference between three and two conducting transistors. Before the signal is propagated to *pMos4* or *nMos4* gate terminal, the other transistors has already transfer the signal



Pattern	d1	d2?	d3?	d4
p0	1	1	0	0
p1	1	0	1	0
p2	0	0	0	0
p3	0	0	0	0

Fig. 12 Two additional DDM columns, Columns d_2 and d_3, possibly exist if the DDM columns of two extreme open defects on a branch are as Columns d_1 and d_4

to the downstream circuit. Therefore, the delay size variation on Branch 11 does not affect the delay size at cell out and the effect of open-defect location can be ignored. Similarly, if *pMos1*, *pMos2*, *pMos3*, *nMos1*, *nMos2*, and *nMos3* are driven by the same input net, open defect on Branch 7 only affect *pMos1* and *nMos1*. The delay size on the driven net is also determined by the performance difference between three and two conducting transistors, and the the effect of open-defect location on Branch 7 can be ignored.

In summary, only for a Type 3 branch, per cell pattern *p*, open on the most upstream segment causes the longest delay while the most downstream segment open causes the shortest delay, as these two opens lead to $max(C_L(p))$ and $min(C_L(p))$, respectively. In a logic test, only when the delay caused by an open defect is larger than the test cycle, the open defect is detectable by the applied test pattern. Given a test frequency, if and only if the delays caused by the two extreme segment opens are both larger or smaller than the test cycle, namely detectable or non-detectable, for identical cell patterns, all opens on the corresponding branch result in identical DDM columns, and are equivalent. Otherwise, we need to identify the multiple different DDM columns caused by the open defects on a branch.

5.4 Open-defect Collapsing

As the delay resulted from the open defects can be accurately identified by simulation only, and we have to simulate at least one open defect per branch, we first collapse the full set to a compact set of open defects by selecting the most upstream segment as the representative open defect per branch. Based on the delay given by the simulator, we derive the delay caused by the most downstream open defect per pattern and then calculate the DDM column of this open. To verify if all opens on a branch equivalent, we compare the two DDM columns of the most upstream respectively downstream segment open.

Typically, the intra-cell branches are very short and transistor gate capacitance is much larger than the interconnect capacitance. Therefore, the variation of the load capacitance due to various open-defect locations on the same branch is expected small.

The open-defect collapsing is described in Algorithm 2. To prevent modeling multiple open-defect locations on the same branch, Algorithm 3 stores in array *branchOpens* with bool values which branches already have been assigned an open defect. In Line 03, this array is initialized to 0. In Lines 04–07, for each branch, open location is identified on the most upstream segment. Next, for all transistors, open locations are identified for the compact set, if not disabled by the user (Line 09). In case a transistor terminal is a fork node, it becomes a transistor-open location that is non-equivalent to any branch-open defect (Line 12). For each of non-fork transistor terminals, if no open-defect location is identified on the branch which includes the transistor terminal yet, we add an open-defect location on this terminal (Lines 13–15).

Algorithm 2 OPEN-DEFECT COLLAPSING

```
01: for all $c \in Cells$ do {
02:   $openCompSet_c = \varnothing$;
03:   for all $b \in Branches_c$ do { $branchOpen[b] = False$; }
04:   for all $b \in Branches_c$ do {
05:     identify the most upstream segment $segm(i,j) \in openFullSet_c$, where $i,j \in b$;
06:     $openCompSet_c = openCompSet_c \cup (i,j)$;
07:     $branchOpen[b] = True$;
08:   }
09:   if $\neg$ disableTrTerminalOpens then {
10:     for all transistors $t$ in $c$ do {
11:       for all $t.tm \in openFullSet_c$ do {
12:         if $t.tm$ is a fork node do {$openCompSet_c = openCompSet_c \cup t.tm$;}
13:         else { if $branchOpen[branch(t.tm)] = False$ then {
14:           $openCompSet_c = openCompSet_c \cup t.tm$;
15:           $branchOpen[branch(t.tm)] = True$;
16: } } } } } }
```

5.5 Open-defect Characterization

Subsequently, we perform open-defect characterization with the compact set, which is described in Algorithm 3. The CAT tool users can set a global resistance value R_{open} for all open defects and a delay size threshold t_{th} (Line 03). t_{th} is used to compared with the delay size caused by each open-pattern combination. If the delay size is larger than t_{th}, the open defect is detected. At each open-defect location, we also can characterize multiple open defects with different resistance values by running additional simulations. For all open defects in the compact set, we apply the set of two-cycle cell patterns and record the delay size per defect-pattern combination at cell output. Subsequently, we determine which two-cycle cell pattern detects which open defect based on the comparison between the cell delay size and t_{th} (Lines 03–07).

Per combination of branch and cell pattern, if the delay size caused by the most upstream segment open is larger than t_{th} while the delay size caused by the most downstream segment open is smaller than t_{th}, the opens in the branch are not equivalent to this cell pattern, and hence cause different DDM columns. Simulation gives the delay size caused by the most upstream segment open, and we calculate the delay size caused by the most downstream open defect based on the load capacitances ratio of the most upstream open to the most downstream open. If only one cell pattern p causes different detection results of the two extreme opens in a branch, only two DDM columns will result from all open defects. In these two DDM columns, both entries per row are identical except in the p row. If more than one cell patterns cause different detection results, we simulate every segment open in the branch to identify all different DDM columns for the best accuracy. For example, the most upstream open defect d_1 in branch b is detected by two patterns p_1 and p_2, but the most downstream open defect d_4 in b is detected by neither p_1 nor p_2. If some intermediate segment open defects can be detected by either p_1 or p_2, four possible DDM columns will be caused by the open defects in b, shown as Figs. 13 and 14. To identify the existence of *d2* and *d3* DDM columns, we run defect characterization.

Algorithm 3 OPEN-DEFECT CHARACTERIZATION

```
01: for all $c \in$ Cells do {
03:   set open-defect resistance $R_{open}$ and $t_{th}$;
02:   for all open $\in$ openCompSet$_c$ do {
04:     for all $p \in$ two-cycle cell patterns$_c$ do {
05:       delay size $t_{max}$ = defect_characterization($c$+open, $p$);
06:       if $t_{max} > t_{th}$ then {
07:         DDM(open, $p$) = 1;
08: } } } }
```

5.6 Open-defect Test Quality Verification and Compensation

After the open-defect characterization, only for each branch in Type 2 described in Section 5.3, we verify if all open defects cause identical DDM columns. If not, we ensure the best test quality by performing additional simulations to identify all different DDM columns.

Quantus Settings 3 [FOR SHORT-DEFECT LOCATION IDENTIFICATION]

```
01: filter_cap -exclude_self_cap False;
02: filter_cap -exclude_floating_nets true;
03: filter_coupling_cap;
04:   -coupling_cap_threshold_absolute 1e-25;
05:   -coupling_cap_threshold_relative 1e-3;
```

Fig. 13 Dedicated Quantus settings for short-defect location identification

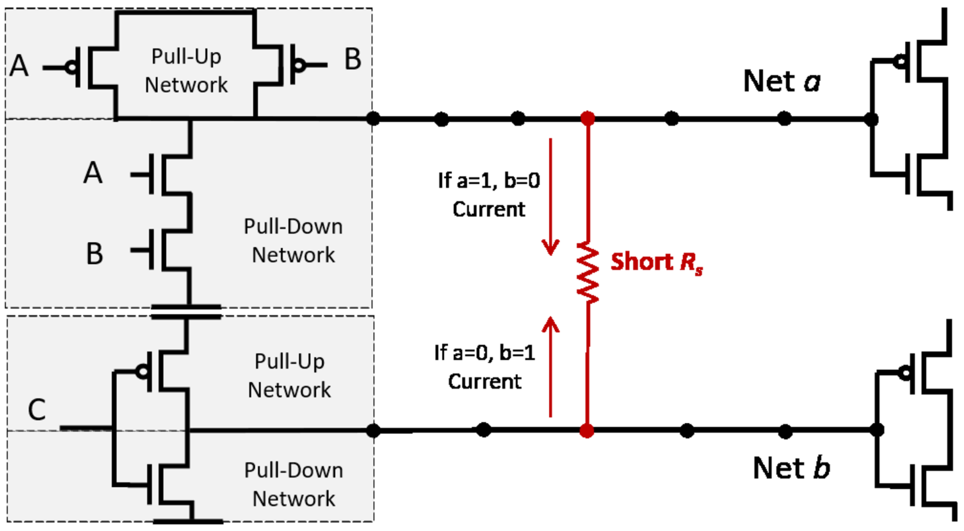


Fig. 14 Short defects between nets without parasitic resistance

The proposed algorithm is described in Algorithm 4. We count the number of cell patterns which cause different detection results for the two extreme open defects per branch (Lines 02–09). If more than one cell pattern causes different detection results, we perform defect characterization for all open defects to identify all different DDM columns caused by these open defects (Lines 10–14). To facilitate the downstream cell-aware ATPG, we collapse the identical DDM columns of the open defects in the same branch into one to save the downstream cell-aware ATPG time (Line 16).

Algorithm 4 OPEN-DEFECT TEST QUALITY VERIFICATION AND COMPENSATION

```
01: for all $c \in Cells$ do {
02:   for all $b \in Branches_c$ do {
03:     $pattern\_counter = 0$;
04:     for all $p \in$ two-cycle cell patterns$_c$ do {
05:       if $t_{max} > t_{th}$ then {
06:         $t_{min} = t_{max} \times (C_{min}(p)/C_{max}(p))$; }
07:         if $t_{min} < t_{th}$ then {
08:           $pattern\_counter$ += 1; }
09:     }
10:     if $pattern\_counter > 1$ then {
11:       for all $p \in$ two-cycle cell patterns$_c$ do {
12:         for all $segm(i, j) \in b$ do {
13:           $d = R_{open} + segm(i, j)$;
14:           $DDM(d, p)$ = defect_characterization($c$+$d$, $p$);
15:     } } }
16:     Collapse identical DDM columns into one for opens on $b$;
17: } }
```

6 Short-defect Identification and Characterization

This section describes how we handle short defects. We first present how we use the PEX tool to assist the short DLI in Section 6.1. Subsequently, we describe the identification of the full set of short-defect locations in Section 6.2. Section 6.3 analyzes the short-defect electrical behaviors to identify the equivalent short defects, which is the basis of short-defect collapsing and test quality verification. The algorithms of short-defect collapsing and characterization are described in Sections 6.4 and 6.5, respectively. Section 6.6 verifies and compensates the short-defect test quality.

6.1 PEX for Short-defect Location Identification

To identify the full set of short-defect locations between intra-cell interconnects, we let the PEX tool extract the parasitic coupling capacitors between all nodes in the library cell; Fig. 13 shows the associated user options for Quantus. All self-coupling capacitors between nodes on the same net are extracted to obtain an accurate simulation model (Line 1). The DLI script will ignore these self-coupling capacitors for short-defect location identification, as we do not want to consider short defects between nodes on a single net. Floating nets are expected not to occur in the heavily optimized circuits that library cells are. However, just in case they do occur any way, the command in Line 2 removes them as a single short connected to floating nets cannot cause a fault. We want a simple cut-off threshold to filter out capacitors for which the nodes which are too far apart in the cell's layout. These far apart nodes pose a low risk to become shorted by a defect that would affect the interconnects. The Quantus MinC function does *not* provide such a straightforward cut-off filter. Hence, we set MinC as low as possible to extract all capacitors and apply our own cut-off threshold for capacitor values in the Modus DLI script. Quantus does not accept MinC=0, and hence we set MinC=10^{-25} Farad below which Quantus does not extract any capacitance value (Lines 3–5).

The capacitance is indicative of the chance that these nets will indeed suffer from a short defect between the two nodes: a large C value indicates that the corresponding two nets run for a significant length parallel at close distance from each other and therefore have an increased probability to be affected by a short defect. Our DLI approach can identify short-defect locations both within a layer as well as in between physically adjacent layers. Note that the number of node pairs grows quadratically with the number of nodes.

6.2 Full Set of Short-defect Locations

For intra-cell interconnects, we guide the PEX tool to extract all parasitic capacitors between net nodes and consider each parasitic capacitor $C(i, j)$ with $conn\_net(i, j) = False$ as a potential interconnect short defect location. For transistors, we model defects inside them as short defects between transistor terminals. Gate-oxide breakdown between gate and source or drain can be exactly modeled by a short resistor between gate and source or drain terminals [22]. Improperly large diffusion area of source and drain due to bad doping may cause a short defect between source and drain [9]. By default, we model transistor-terminal shorts between *gate-source, gate-drain, source-drain* [8, 9, 31]. The parasitic capacitors between interconnects and the pairs of transistor terminals constitute the full set of short-defect locations per cell.

Modus' DLI function has the following user controls with which the user can authorize the tool to filter out certain short locations.

- Parameter *disableTrTerminalShorts* allows to explicitly disable the identification of short-defect locations on transistor terminals (default: *false*).
- *TrTerminalShortSet* specifies which set of transistor terminal pairs are considered as possible short locations (default: {*gate-source, gate-drain, gate-bulk, source-drain*}).
- A *blocklist* specifies layer pairs between which shorts should *not* be considered. This list is technology dependent. Advanced CMOS technologies with feature sizes below 10nm typically have more layers in their library cells and some of these layers are sensitive for inter-layer shorts [4] (default: empty list).
- Threshold value C_{th} defines that extracted parasitic capacitors will not be identified as short-defect locations if $C < C_{\text{th}}$ (default: $C_{\text{th}} = 0$).

Algorithm 5 FULL-SET OF SHORT-DEFECT LOCATION IDENTIFICATION

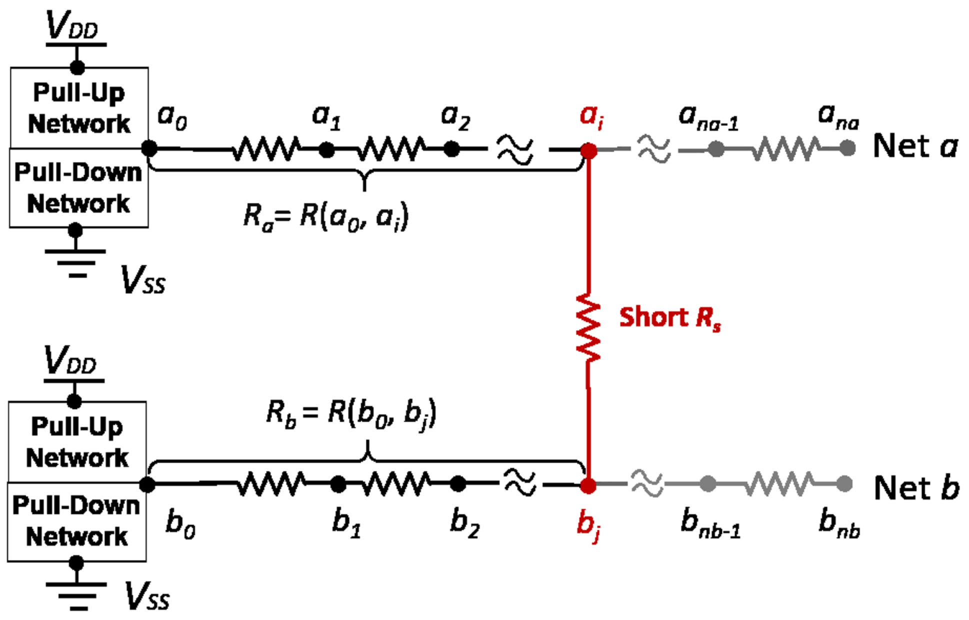
```
01: for all $c \in Cells$ do {
02:   $shortFullSet_c = \varnothing$;
03:   for all $a, b \in Nets_c$ with $a \neq b$ do {
04:     for all $i \in a$ and $j \in b$ do {
05:       if $((lyr(i), lyr(j)) \notin BlockList) \wedge (C(i,j) \geq C_{th})$ then {
06:         $shortFullSet_c = shortFullSet_c \cup (i,j)$;
07:   } } }
08:   if $\neg$ disableTrTerminalShorts then {
09:     for all transistors $t$ in $c$ do {
10:       for all transistor terminal pairs $(tm_1, tm_2) \in TrTerminalShortSet$ do {
11:         $shortFullSet_c = shortFullSet_c \cup (tm_1, tm_2)$;
12: } } } }
```

The DLI operation to identify the full set of short-defect locations within a library cell is described in Algorithm 5. In Line 02, for cell c, the full set of short-defect locations is initialized. Subsequently, the parasitic Cs are evaluated to identify locations for short defects between intra-cell nets (Lines 03–07). For each net pair $a, b \in Nets_c$, if the node pair of $i \in a$ and $j \in b$ is not on the blocklist and for which holds $C(i,j) \geq C_{th}$, we add it to the full set (Lines 05–06). Next, for all transistors, short-defect locations between the terminals of *TrTerminalShortSet* are identified, only if not disabled by the user (Lines 08–12).

6.3 Short-defect Equivalence

A short defect can happen between any two nodes from two disjoint Nets a and b and cause the whole net-pair shorted together, shown in Fig. 14 which is a partial schematic of a full adder cell in Cadence's GPDK045 library. To test a short defect between a and b, we should first activate the short defect by assigning opposite logic values on a and b, respectively. As each cell-internal net is driven by either pull-up or pull-down network, logic 1 on a net is obtained by switching on the pull-up network to connect the net to the power net V_{DD},

Fig. 15 Short defects between nets with parasitic resistance



while logic 0 is from the ground net V_{SS} through the pull-down network. When a static signal is applied on a net, no current flows on the net as the logic signal is used to control downstream transistor gates that are considered as high impedance terminals. Therefore, the activation consequence is that a conductive path from V_{DD} to V_{SS} results from the short defect, shown in Fig. 14. If the interconnects are ideal, namely without any parasitic resistance, the identical logic values on both nets are interpreted from the voltage division between the transistor networks driving the two shorted nets, assuming the short-defect resistance is small and can be ignored. If the short-defect resistance increase, we should also consider the voltage drop on the short defect increases, but the analysis in this section is still valid. No matter which two nodes from Net *a* respectively *b* are shorted and how big R_{short} is, the voltage value at the short location is fixed. All short defects between the two nets cause the same electrical effect and hence are equivalent. Please note that it is possible that multiple cell patterns activate a short defect. In Fig. 14, logic 1 on Net *a* and logic 0 on Net *b* can be obtained by three combinations of inputs A, B, and C: {A=0,B=1,C=1; A=1,B=0,C=1; A=0,B=0,C=1}. As different input signal combinations switch on different transistors, the voltage division between the transistors varies with the inputs. Per input combination, the voltage values at all short-defect locations are identical.

Considering the parasitic resistance, per input combination, the voltage values at different short-defect locations are slightly different. We build a short defect model including the extracted parasitic resistors on nets, shown in Fig. 15. To make the analysis easily to understand, we assume R_{short} approximate to 0Ω, namely the voltage values at two shorted nodes are equal. The numbers of nodes on Nets *a* and *b* are *na* and *nb*, respectively. The short defect can be between node a_i on Net *a* and node b_j on Net *b* if $C(a_i, b_j)$ exists in the extracted DSPF netlist, where $i \in [0..na]$ and $j \in [0..nb]$. The logic value at a short-defect location is determined by the voltage division between two parts resistances on the two sides of the location. In Fig. 15, the two parts of resistances are $(R_{pull\text{-}a} + R_a)$ and $(R_{pull\text{-}b} + R_b)$, where $R_{pull\text{-}a}$ and $R_{pull\text{-}b}$ are the resistances of switched-on transistors in the pull networks driving Nets *a* and *b*, respectively; R_a and R_b are the Net *a* respectively Net *b* parasitic resistances involved in the conductive path from V_{DD} to V_{SS} due to the short defect. R_a is the sum of resistances from node a_0 till node a_i which is the short location node. Similarly, R_b is the sum of resistances from node b_0 till the short location b_j. A cell pattern causing opposite logic values on Nets *a* and *b* determines the conducted transistors in the pull-up and pull-down networks, and therefore determines the network resistances $R_{pull\text{-}a}$ and $R_{pull\text{-}b}$. R_a and R_b are determined by the short-defect location.

Given a cell pattern *p* activating a short defect between Nets *a* and *b*, the voltage values V_a and V_b at the shorted node a_i respectively b_j vary with the locations of a_i and b_j, as their locations affect R_a and R_b. If the voltage variation makes the interpreted logic value on the nets change between 0 and 1, two groups of equivalent short defects exist between the nets for *p*.

Typically, the parasitic resistance of interconnects is small, and hence the voltage drop on parasitic resistors is much smaller than that on transistors. As source-drain voltage of transistors does not significantly change with the variation of the short location, we assume the transistors in the shorted path can be seen as linear resistors [30]. With this assumption, the larger R_a and the smaller R_b, the more voltage drop on Net *a*. Therefore, it is more probable that the interpreted logic values at node a_0 to node a_i are different. When R_a is maximized and R_b is minimized by shorting node a_{na} and b_0, shown in Fig. 16a, the voltage division is determined by the ratio of

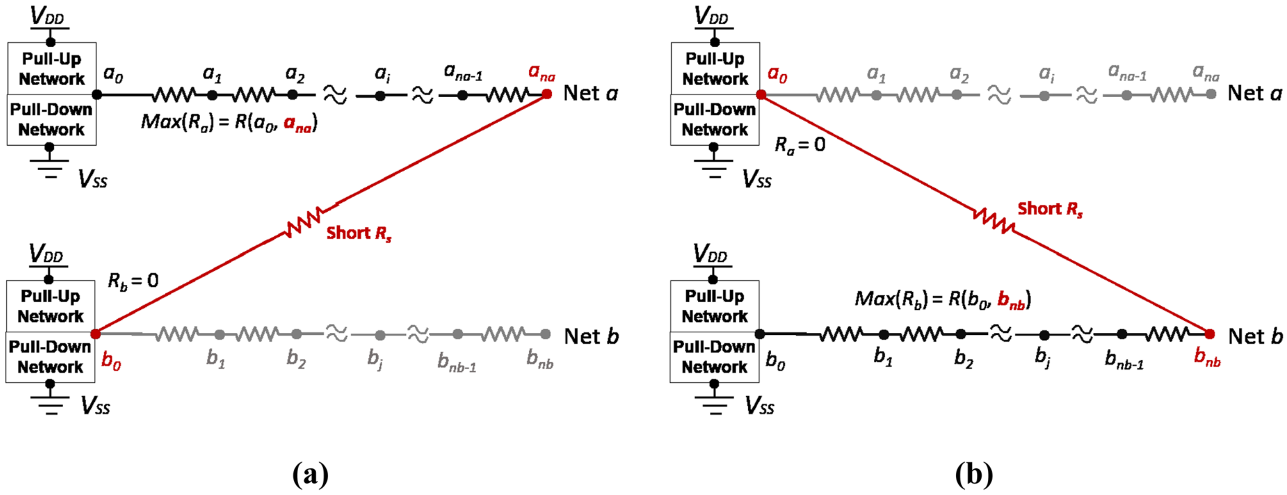
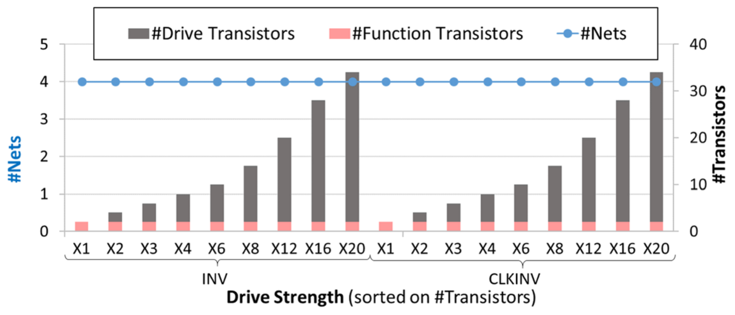
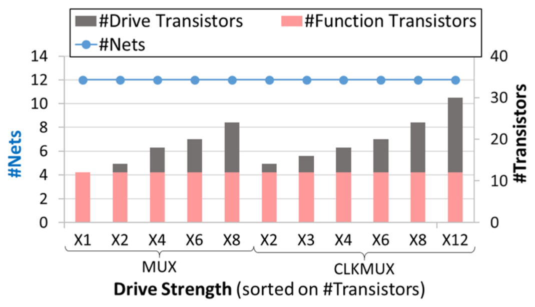


Fig. 16 Two extreme short-defect locations lead to the highest probability of that **(a)** Net *a* is dominated and **(b)** Net *b* is dominated



(a) GPDK045 inverter cells.



(b) GPDK045 multiplexer 2-to-1 cells.

Fig. 17 The number of transistors but not nets increases with the drive strength

($R_{pull\text{-}a} + max(R_a)$) to $R_{pull\text{-}b}$. If a pull-up network drives Net a and a pull-down network drives Net b, the short defect in Fig. 16a leads to the minimum V_a and V_b which the most probably is interpreted into logic 0, which means the logic value on Net a is forced to change from logic 1 to 0. If a pull-down network drives Net a and a pull-up network drives Net b, as the voltage drop on Net a is maximized and the voltage drop on Net b is minimized to zero, the short defect in Fig. 16a leads to the maximum V_a and V_b, namely leads to the highest probability that logic values on a and b are 1 and a is still dominated by b. Similarly, Fig. 16b shows the short-defect location which causes the highest probability that Net a dominates Net b. In summary, between Net pair [a, b] the two diagonal short defects shown in Fig. 16a and Fig. 16b determine the range of voltages at short-defect locations for every cell pattern activating the short defects between Nets a and b. If for all cell patterns, the voltage values at the two diagonal short-defect locations are interpreted into identical logic values, all short defects between a and b are equivalent.

Power net V_{DD} and ground net V_{SS} are voltage sources, and no transistor network drives them. Assuming a short defect happens between V_{DD}/V_{SS} and some other net, the voltage at the short-defect location is determined by the voltage drop on the parasitic resistance of V_{DD} $/V_{SS}$. Typically, the parasitic resistance on V_{DD} or V_{SS} is up to several decades ohm, while the transistor network resistance is at least several hundreds ohm. The voltage drop on V_{DD} or V_{SS} is much smaller than that on the other side of the short defect, and the change of short-defect location cannot lead to a drastic voltage variation at the short-defect location. Therefore, the voltage values at the short-defect locations are always close to the value on V_{DD} or V_{SS}, namely V_{DD} or V_{SS} always dominates the other net driven by transistors.

In summary, the short-defect location affects the voltage value on the shorted nets through the parasitic resistance involved into the shorted path. Typically, transistor network resistances are much larger than the parasitic resistances, and therefore varying the short-defect location cannot make the logic value on the shorted nets change, namely all short defects are equivalent. However, in some specific situations, varying the short-defect location result in some non-equivalent defects that are discussed later. The range of voltages at the short-defect location is determined by the two diagonal short locations. If the two diagonal short defects between a net pair are equivalent, all shorts between the net pair are equivalent.

6.4 Short-defect Collapsing

Based on the analysis of short-defect equivalence, for many net pairs, the short defects are equivalent. To minimize the defect count to be simulated, we collapse the full set of short defects into a compact set by identifying only one short-defect location and simulate the compact set. Afterward, we identify the net pairs with non-equivalent short defects and identify the DDM columns for all these short defects to compensate the test quality. This location should meet two conditions: (1) the location is based on an extracted parasitic capacitor and (2) compared with the other extracted capacitors, the node pair of this capacitor leads to on one net the maximum parasitic R while on the other net the minimum parasitic R involved into the shorted path from V_{DD} to V_{SS}, and therefore results in either boundary of the voltage variation due to the short location.

The short-defect collapsing algorithm is described in Algorithm 6. As we want to identify at most one short per net pair, we use array *netShorts* to store which net pairs already have been assigned a short defect; in Line 03, this array is initialized. Subsequently, for each net pair $a, b \in Nets_c$ which does not have a short identified yet (Line 05), we identify the node pair $i \in a$ and $j \in b$ in the full set of short-defect locations and leading to the maximal parasitic R of Net a and minimal R of Net b that are involved into the shorted path (Lines 08–13). Next, for all transistors, if the user does not disable the transistor terminal shorts, only the transistor terminal pairs between which there is not already a short between the two corresponding nets are added to the compact set (Lines 15–20). If the transistor terminal leads to larger difference of the involved parasitic resistances, we replace the interconnect short location by the transistor-terminal pair (Lines 21–25).

6.5 Short-defect Characterization

Defect characterization is performed with only one short defect per net pair. To verify the test equivalence of short defects, except the DDM entry, we record more simulation results: per combination of short defect and cell pattern, voltage value V_p at the selected short-defect location and the current I_p in the conductive path from V_{DD} to V_{SS}.

Algorithm 6 SHORT-DEFECT COLLAPSING

```
01: for all $c \in Cells$ do {
02:   $shortCompSet_c = \varnothing$;
03:   for all $a, b \in Nets_c$ with $a \neq b$ do { $netShort[a, b]$ = []; }
04:   for all $a, b \in Nets_c$ with $a \neq b$ do {
05:     if $netShort[a, b]$= 0 then {
06:       $mi$ = []; $mj$ = []; $R_{mi}$ = 0; $R_{mj}$ = 0
07:       for all $i \in a$ and $j \in b$ do {
08:         if $C(i, j) \in shortFullSet_c$ then {
09:           if $(R_i - R_j) > (R_{mi} - R_{mj})$ then {
10:             $R_{mi} = R_i$; $R_{mj} = R_j$; $mi = [i]$; $mj = [j]$;
11:       } } }
12:       add node pair $(mi, mj)$ to $shortCompSet_c$;
13:       $netShort[a, b]$ = [mi,mj]; $netShort[b, a]$ = [mi,mj];
14:   } }
15:   if ¬ $disableTrTerminalShorts$ then {
16:     for all transistors $t$ in $c$ do {
17:       for all transistor terminal pairs $(tm_1, tm_2) \in TrTerminalShortSet$ do {
18:         if $netShort[net(tm_1),(net(tm_2)]$ = 0 then {
19:           add transistor terminal pair $(tm_1, tm_2)$ to $shortCompSet_c$;
20:           $netShort[net(tm_1),net(tm_2)]$ = 1; $netShort[net(tm_2),net(tm_1)]$ = 1;
21:         else {
22:           $i$ = $netShort[net(tm_1),(net(tm_2)][0]$;
23:           $j$ = $netShort[net(tm_1),(net(tm_2)][1]$;
24:           if $R_{tm_1} - R_{tm_2} > R_i - R_j$ then {
25:             replace $(i, j)$ by transistor terminal pair $(tm_1, tm_2)$ in $shortCompSet_c$;
26: } } } } } } }
```

Algorithm 7 SHORT-DEFECT CHARACTERIZATION

```
01: for all $c \in Cells$ do {
02:   set short-defect resistance $R_{short}$;
03:   for all $s_1 \in shortCompSet_c$ do {
04:     for all $p \in$ $one\text{-}cycle\ cell\ patterns_c$ do {
05:       $Voltage(s_1, p)$, $Current(s_1, p)$, $DDM(s_1, p)$ = $Simulation(c+s_1, p)$;
06: } } }
```

6.6 Short-defect Test Quality Verification and Compensation

To check if all short defects per net pair are equivalent, we should know for each cell pattern activating the short defects, if the two diagonal short defects are equivalent. Based on the simulation result of the selected short defect per net pair, we identify the cell patterns for which the short defects between a net pair are possible to be non-equivalent, and hence cause different resulting DDM entries. Afterward, we simulate the other diagonal short defect for the net pair with only these suspicious cell patterns. If the two diagonal short defects are non-equivalent, we perform additional simulations for all short defects between the net pair to obtain all different DDM columns to maintain the test quality.

With the voltage and current values at the location of each simulated short defect, we calculate V_p/I_p which is the sum of the resistance of pull-down network and parasitic resistance on the corresponding driven net. Similarly, $(V_{DD} - V_p)/I_p$ is pull-up network resistance with the parasitic resistance on the corresponding driven net. The involved parasitic resistance on two shorted nets are calculated by processing the extracted DSPF netlist. Therefore, we calculate the transistor-network resistances by subtracting the parasitic resistances in V_p/I_p and $(V_{DD} - V_p)/I_p$. Considering the effect of short-defect location, we classify the relationships between the transistor and parasitic resistances into four cases per cell pattern p that activates the short defect.

Case 1: $(R_{pull-a}(p)/R_{pull-b}(p) > 3/2) \wedge (R_{pull-a}(p) > 10 \times max(R_a)) \wedge (R_{pull-b}(p) > 10 \times max(R_b))$ If no parasitic resistance is involved in the conductive path caused by a short defect, $R_{pull-a}/R_{pull-b} > 3/2$ results in more than 60% voltage drops on the transistor network driving Net a. If a pull-up network drives Net a, the voltage at the short location V_p is smaller than $40\%V_{\mathrm{DD}}$. The logic values on Nets a and b are 0. If a pull-down network drives Net a, V_p is larger than $60\%V_{\mathrm{DD}}$. The logic values on a and b are interpreted into 1. Therefore, Net b dominates Net b. Furthermore, transistor resistances are much larger than the maximum of R_a and R_b. The variations of R_a and R_b due to short-defect location cannot cause the voltage value at the short-defect location to vary between logic 0 and 1. The effect of short-defect location can be ignored, and hence all short defects between a and b are equivalent for p.

Case 2: $(R_{pull-a}(p)/R_{pull-b}(p) < 2/3) \wedge (R_{pull-a}(p) > 10 \times max(R_a)) \wedge (R_{pull-b}(p) > 10 \times max(R_b))$ Case 2 is similar to Case 1. The voltage value at the short-defect location is also either smaller than 40% $40\%_{\mathrm{DD}}$ or larger than $60\%V_{\mathrm{DD}}$, and the effect of short-defect location can be ignored and all short defects are equivalent for pattern p. But in this case, Net a dominates Net a.

Case 3: $2/3 \leqslant R_{pull-a}(p)/R_{pull-b}(p) \leqslant 2/3$ If no parasitic resistance is involved in the conductive path caused by a short defect, the voltage value at the short-defect location is in the range of $[40\%V_{\mathrm{DD}}, 60\%V_{\mathrm{DD}}]$. The threshold voltage between logic 0 and 1 is in this range for the CMOS technology. A small voltage variation can change the interpreted logic value. Therefore, the variations of R_a and R_b may cause different fault effects. In this case, two possible groups of equivalent short defects exist between the net pair $[a, b]$. One group causes the logic values on both nets to be 1, while the other group causes both nets' logic signals to be 0.

Case 4: $(max(R_a) \geqslant 10\%R_{pull-a}(p)) \vee (max(R_b) \geqslant 10\%R_{pull-b}(p)) \vee (max(R_a) \geqslant 10\%R_{pull-b}(p)) \vee (max(R_b) \geqslant 10\%R_{pull-a}(p))$ In standard cells with high drive strength, multiple parallel transistors are controlled by the same input signal. When these parallel transistors are switched on simultaneously, the signal on their driven net is pulled up or down by these parallel transistors together, and hence the drive strength increases. However, resistances of the transistor networks offering high drive strength decrease due to the parallel conducted transistors. If the transistor-network resistances are compatible with the parasitic resistance, the short-defect location may influence the logic value on the shorted Nets a and b. For the net pairs of which parasitic resistances are larger than 10% transistor-network resistance, we check if the two diagonal short defects are equivalent.

If Case 1 or 2 happens for a cell pattern, all short defects between the corresponding net pair are equivalent. Only in Cases 3 and 4, test quality verification and compensation is required to maintain the test quality. The process is described in Algorithm 7. For each net pair, we initiate Set *nonEquPatterns* to store the cell patterns for which the short defects are non-equivalent (Line 03). Based on the voltage value at and current value through each short-defect location, we calculate the transistor-network resistances per cell pattern p that activates the short defect (Lines 04–13). For the cell pattern with which the effect of short-defect location cannot be ignored, we simulate the other diagonal short defect which identifies the range of voltage values with the already simulated diagonal short defect together. Only if the two DDM entries of the two diagonal short defects are different, short defects between a and b are non-equivalent for pattern p, and we store p in the set *nonEquPatterns* (Lines 18–24). If for only one cell pattern, the short defects are non-equivalent, we derive the only two different DDM columns in which the entries are all identical except in pattern p row. (Lines 26–31). Otherwise, we simulate all short defects only with the identified cell patterns in *nonEquPatterns* to get all different DDM columns (Lines 33–36). To facilitate the downstream ATPG step, we collapse all identical DDM columns into one (Line 38).

Algorithm 8 SHORT-DEFECT TEST QUALITY VERIFICATION AND COMPENSATION

```
for all $c \in Cells$ do {
  for all $a, b \in Nets_c$ with $a \neq b$ and short defect $s1$ between $a$ and $b$ do {
    $nonEquPatterns = \{\}$;
    for all $p \in P_c$ do {
      if $a$ is driven by a pull-down network $\wedge$ $b$ is driven by a pull-up network then {
        $R_{pull\text{-}a}(p) = (Voltage(s1, p)/Current(s1, p))$ - $R_a$;
        $R_{pull\text{-}b}(p) = (V_{DD} - Voltage(s1, p))/I_p$ - $R_b$ - $R_{short}$;
      }
      else if $a$ is driven by a pull-up network $\wedge$ $b$ is driven by a pull-down network then {
        $R_{pull\text{-}a}(p) = (V_{DD} - Voltage(s1, p))/I_p$ - $R_a$ - $R_{short}$;
        $R_{pull\text{-}b}(p) = (Voltage(s1, p)/Current(s1, p))$ - $R_b$;
      }
      else continue;
      if $(R_{pull\text{-}a}(p)/R_{pull\text{-}b}(p) < 2/3 \vee R_{pull\text{-}a}(p)/R_{pull\text{-}b}(p) > 3/2)$
        $\wedge$ $(R_{pull\text{-}a}(p) > 10 \times max(R_a) \wedge R_{pull\text{-}b}(p) > 10 \times max(R_b))$ then {
        all short defects between Nets $a$ and $b$ equivalent for $p$; continue;
      }
      else {
        identify the other diagonal node pair $(i \in a, j \in b)$ with maximum $(R_b - R_a)$;
        inject short defect $s2$ with resistance $R_{short}$ between $(i, j)$;
        $DDM(s2, p) = Simulation(c{+}s2, p)$;
        if $DDM(s1, p) \neq DDM(s2, p)$ then {
          short defects between $a$ and $b$ are non-equivalent for $p$;
          $nonEquPatterns = nonEquPatterns \cup p$;
    } } }
    if $|nonEquPatterns| = 1$ then {
      for all pattern $p \in$ one-cycle cell patterns$_c$, do {
        if $p \in nonEquPatterns$ then {
          $DDM(s2, p) = \neg DDM(s1, p)$;
        else {$DDM(s2, p) = \neg DDM(s1, p)$; }
      add new DDM columns of $s2$ to DDM;
    } } }
    if $|nonEquPatterns| > 1$ then {
      for all short $s$ between $a$ and $b$, do {
        for all pattern $p \in nonEquPatterns$, do {
          $DDM(s, p) = Simulation(c + s, p)$;
    } } }
    collapsing all identical DDM columns into one column;
} } }
```

7 Experimental Results

The experiment results reported in this paper are based on 351 combinational cells in Cadence's GPDK045 45nm library [3]. This library is planar CMOS technology and all cells contain four layers: n-diffusion, p-diffusion, poly, and metal-1 layers. The EDA tool versions used in our experiments are Cadence Pegasus v18.1, Quantus Extraction Solution v18.1, and Modus v20.1. For the full set DLI, we use the default user settings which includes all possible defect locations, such that guarantees the best test quality; for defect characterization, we set open-defect resistance 1GΩ and short-defect resistance 0.001Ω. The tolerated delay threshold value is set to 1ns.

7.1 GPDK045 CMOS Library

A standard-cell library contains two types of cells: logic cells and physical only cells. Physical only cells, such as tie-up, filler, and decap cells, have no inputs or outputs, hence we cannot perform the logic test that requires applying test stimuli on the inputs and observing responses on the outputs. CAT, as a logic test method, is applied on only logic cells. Typically, a logic cell with base drive strength that is denoted by 'D1' or 'X1' in the cell name contains only transistors that implement the cell's logic function. We refer to these transistors as *function transistors*. Designers increase the drive strength of cells by adding so-called *drive transistors* in parallel with the function transistors driving the cell outputs. Therefore, both function and drive transistors are switched on or off by the same input signal and contribute to pull up or pull down the output simultaneously, which increases the drive strength. For a group of cells with the same logic function, the variation of drive strength influences the number of transistors but not the number of nets, which indicates the number of short-defect locations in the compact set is independent of the drive strength. For example, Fig. 17 shows the drive strengths and the number of transistors and nets in all GPDK045 inverter and multiplexer 2-to-1 cells. The number of nets always stays the same for all inverter or multiplexer cells, but the number of transistors increases with the drive strength.

The GPDK045 cell information is shown in Fig. 18. The cells with the most inputs are AOI33, AOI222, OAI33, OAI222, and MUX4-to-1. As they both have six inputs and one output, the number of one-cycle cell patterns is $2^6 = 64$. MUX4-to-1 cell requires the most (i.e. 128) two-cycle cell patterns in which one input signal change causes an output transition.

7.2 Example Cell AND2X1

First, we illustrate our approach with cell AND2X1. The AND2X1 cell has two inputs, A and B, one output Y, and performs a logic-AND function. Three I/Os, power and ground connections, and six transistors with four terminals give a total of 29 terminals for this cell. As shown in Fig. 19a, these terminals are interconnected by seven nets: $\{A, B, Y, V_{DD}, V_{SS}, n1, n2\}$.

Figure 19b, shows the layout of AND2X1. The PEX tool partitions the seven nets into 43 segments excluding eight virtual segments, each with its own parasitic resistor. Figure 19c illustrates the PEX partitioning of nets into segments for Net A, which interconnects input terminal A and the gate terminals of Transistors $Mp0$ and $Mn1$. PEX introduces four internal nodes on Net A. Nodes $A\#9$ and $A\#4$ mark the transition points between respectively metal-1 and poly-silicon to the contact between these two layers. At Node $A\#3$, the net forks out to the PMOS and NMOS transistors. Node $A\#5$ is introduced to mark the 90° bend point in the 'L'-shape. The dangling segments at the extreme ends of the poly-silicon and metal-1 structures are not reported. Example Net A consists of six segments, partitioned over three branches: $\{A, A\#9, A\#4, A\#3\}$, $\{A\#3, Mp0\#g\}$, and $\{A\#3, A\#5, Mn1\#g\}$. For each of the three branches, we identify the segment on

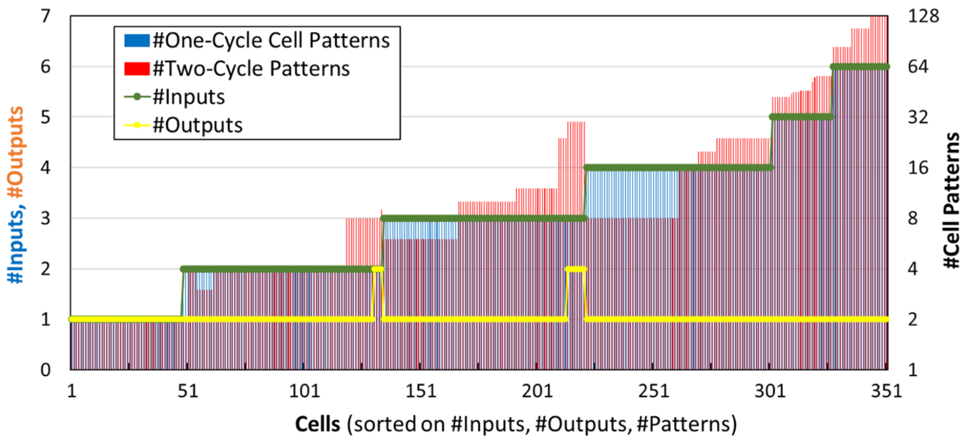


Fig. 18 The number of library cells' inputs, outputs, and all possible test patterns

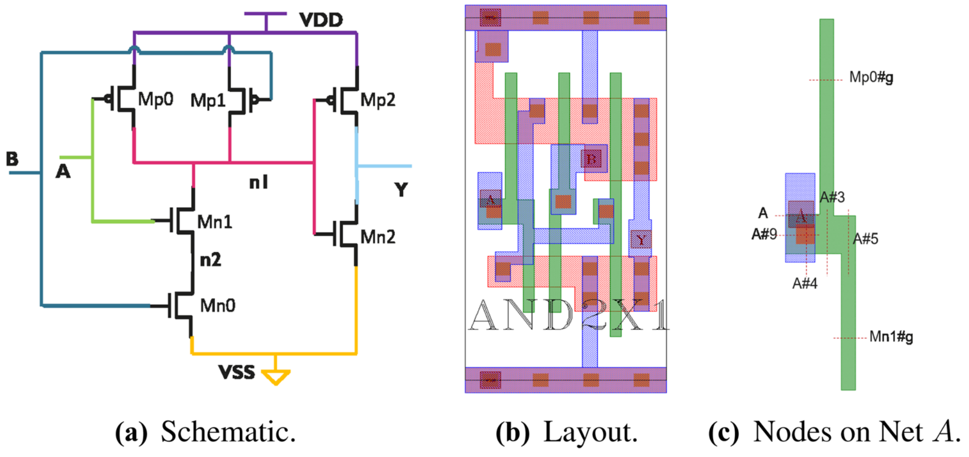


Fig. 19 GPDK045 library cell AND2X1

which an open is the most likely to happen. Opens on the latter two branches are equivalent to transistor gate-terminal opens on $Mp0\#g$ and $Mn1\#g$. Therefore, these two gate-terminal opens are not selected into the compact set.

For the complete cell AND2X1, identifying open-defect locations on *all* segments and on *all* gate, source, and drain transistor terminals results in 43+18 = 61 open-defect locations. However, AND2X1 has 20 branches; we collapse the open-defect locations on 43 segments into on 20 branches. Afterwards, only seven transistor-terminal opens are not on any physical branch. So, only 20 branches and three transistor-terminal opens out of 61 (= 37.7%) open-defect locations are selected for the compact set. Among the 20 branch opens, nine opens cause either no delay or delay longer than 1 μs (= 1000ns) for all cell patterns. As opens on these nine branches cause delay much larger than the detection threshold of delay size defined by the user, and the load capacitance variation on a branch cannot be 1000×, all opens on these branches will cause delay larger than 1ns and hence be detected by the same patterns, namely are equivalent. We do not verify test quality for these branches. The other eleven open defects causing delay smaller than 1 μs and larger than 1ns are shown in Table 1. The column `Nominal Delay' presents the delays for all cell patterns under the defect-free situation. Based on the load capacitances after opens per branch, we calculate the delay size caused by the most downstream segment open on each of the eleven branches, shown in Table 2. All the calculated delay sizes in Table 2 are larger than 1ns. Therefore, for all branches, the most upstream and the most downstream segment opens are detected by the same set of cell patterns. All opens on the branches are equivalent, and no additional simulation to be performed.

For Branches on which {d3, d3′}, {d4, d4′}, {d5, d5′}, {d6, d6′}, {d8, d8′}, {d9, d9′} are, the delay sizes caused by the most upstream and the most downstream segment opens are identical. The reason can be either the branch contains only one segment, or the load capacitance variation caused by open-defect location is too small to impact the delay size.

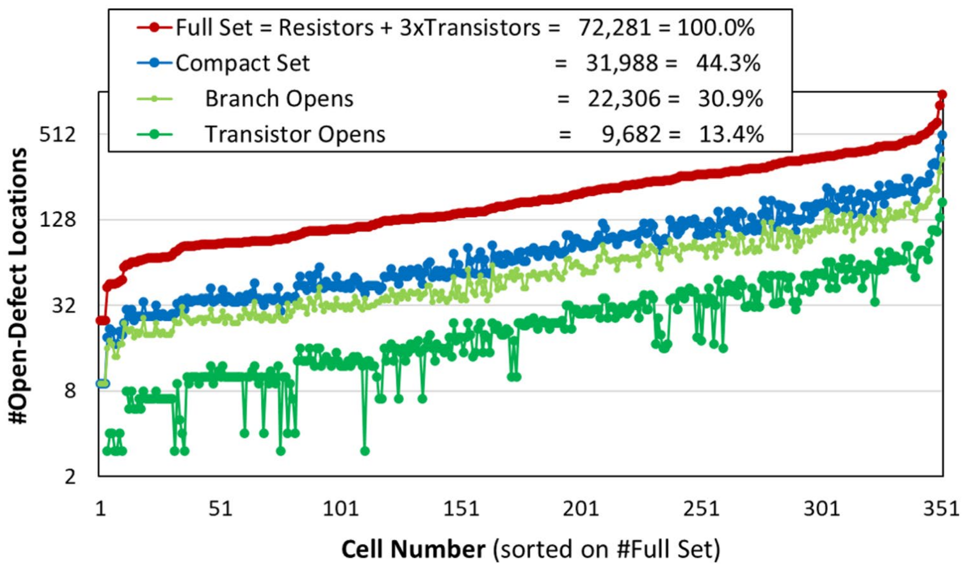


Fig. 20 Full vs. compact sets of open-defect locations in GPDK045 library cells and the division of the latter into branch and transistor opens

Table 1 Delay sizes caused by the most upstream segment open per branch

Pattern		Two-Cycle Detection: Delay Matrix (unit: ns)										
AB;AB/Y	Nominal Delay	$d1$	$d2$	$d3$	$d4$	$d5$	$d6$	$d7$	$d8$	$d9$	$d10$	$d11$
p0 = 01;11/H	0.050	49.19		33.09	62.46		116.16	503.76	8.96			799.67
p1 = 10;11/H	0.052		54.30	33.09		60.26	116.07		31.16		607.03	843.77
p2 = 11;01/L	0.037			62.99	70.01		29.18	479.99				834.70
p3 = 11;10/L	0.039			62.95		71.16	32.11			26.20	593.98	880.41

Table 2 Calculated delay sizes caused by the most downstream segment open per branch

Pattern		Two-Cycle Detection: Delay Matrix (unit: ns)										
AB;AB/Y	Nominal Delay	$d1'$	$d2'$	$d3'$	$d4'$	$d5'$	$d6'$	$d7'$	$d8'$	$d9'$	$d10'$	$d11'$
p1 = 01;11/H	0.050	39.61		33.09	62.46		116.16	395.23	8.96			666.46
p2 = 10;11/H	0.052		36.83	33.09		60.26	116.07		31.16		459.44	703.21
p3 = 11;01/L	0.037			62.99	70.01		29.18	376.59				695.65
p4 = 11;10/L	0.039			62.95		71.16	32.11			26.20	449.56	733.75

Next, we consider short-defect locations. Cell AND2X1 has 58 nodes, which make for $\binom{58}{2} = 1{,}653$ node pairs. Subtracting the 257 node pairs for which both nodes are part of the same net leaves 1,396 node pairs between disjoint nets; this is the theoretical maximum for the number of short-defect locations. The PEX tool extracts only 745 parasitic capacitors above its user-specified PEX minimum of 10^{-25} F. The DLI function considers at most one short-defect location per net pair. Cell AND2X1 has six physical nets, except Net *n2* connecting $Mn0\#d$ and $Mn1\#s$ between which there is no real physical interconnection but just a shared diffusion area. Therefore, we consider $\binom{6}{2} = 15$ net pairs. As they all have extracted parasitic capacitors, for each one we identify one representative short-defect location by selecting the capacitor with the maximum value. For each transistor, identifying short-defect locations between three terminal pairs would result in 18 shorts. However, some transistor-terminal short locations are between the same net pairs, hence are equivalent. Only 15 terminal shorts are non-equivalent to each other and ten of them are equivalent to the net-pair shorts already identified based on the parasitic capacitors. Therefore, we only need to add five more transistor-terminal short locations into the compact set. Furthermore, three of these shorts (*A*-*Y*, *B*-*Y*, and *A*-*B*) are equivalent to the *port-shorts* at the cell boundary. In total, 20 of the 21 potential net-pair short locations are considered realistic in our approach; this constitutes a reduction of 97.4% from the 745+24 = 769 possible short locations based on the extracted netlist. The corresponding simulation time is reduced from 74 to only 7 seconds, which is a 90.5% reduction. The 20 identified cell-internal short-defect locations also cover the six stuck-at faults at the cell boundary, i.e., Ports *A*, *B*, and *Y* either stuck-at-zero or -one.

Table 3 shows the voltages at short-defect locations given by the simulation. The power voltage value is set to be 1.1v based on the library Liberty files. Cell patterns are in the first column. The second to the last columns in the second row give the names of net pairs between which we inject a diagonal short defect. The shorted net pairs of which one net is V_{DD} or V_{SS} are not listed in Table 3 as V_{DD} or V_{SS} always strongly dominates the other net. It is not necessary to verify the short-defect equivalence for these net pairs. The highlighted items in Table 3 indicate the combinations of net pair and cell pattern may have multiple detection results with changing the short-defect location, because the voltage value at the short-defect location is close to the threshold of logic 1. For all the other net pairs, the voltage values at short-defect location are strong logic ‘1’ or ‘0’ and their parasitic resistances are much smaller

Table 3 Calculated delay sizes caused by the most downstream segment open per branch

Pattern	One-Cycle Detection: Voltage Matrix (unit: v)					
AB/Y	**A-B**	**A-Y**	**B-Y**	**A-n1**	**B-n1**	**Y-n1**
p0 = 00/L	0	0	0	0.16	0.15	0.89
p1 = 01/L	0.48	0	0.49	0.08	1.1	0.81
p2 = 10/L	0.47	0.87	0	1.1	0.07	0.81
p3 = 11/H	1.1	1.1	1.1	1.04	1.05	0.34

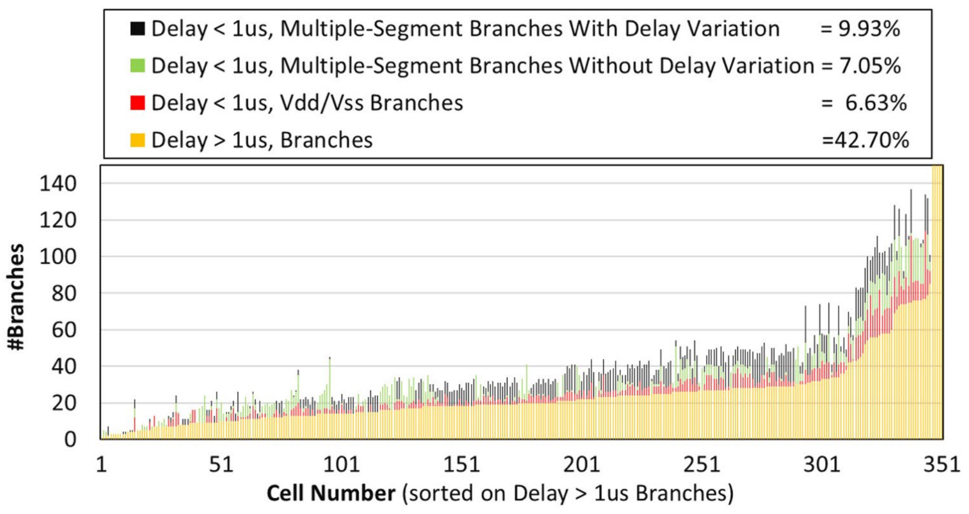


Fig. 21 The classification of the open branches causing delays larger than 1ns

than the driving transistor networks. Therefore, the short-defect location does not impact the detection results. Only for net pair *A*-*B*, we simulate the other diagonal short defect with only cell patterns '01/L' and '10/L'. The resulting DDM columns of the two diagonal shorts are identical, so all short defects between *A* and *B* will cause identical DDM columns, namely are equivalent. The test quality is the same as that based on full set of defects. We do not need to perform additional simulation.

7.3 45nm Technology Library

The PEX run on all 351 combinational cells of the GPDK045 library [3] resulted in 6,438 transistors, 52,967 extracted net segments, and 1,278,098 extracted parasitic capacitors, and took 11.0 hours. Note that there are 24.1× more parasitic *C*s extracted than parasitic *R*s. The simulation time for short-defect characterization dominates the total time cost.

Assigning open defects to all net segments and to three terminals for each transistor results in 52,967 + 3 × 6,438 = 72,281 open locations in the full sets in which 31,988 open defects are selected into the compact set for the downstream defect characterization. This constitutes a reduction of 55.7%. The DLI results per library cell for opens are shown in Fig. 20.

The simulation time of the compact set is 26.6 hours, which indicates the simulation based on the full set would take around 67 hours. For the total 22,306 branches, 74.6% are branches of which the most upstream segment opens are detected. Subsequently, we verify the test quality for the 74.6% branches. The other 25.4% branch opens cause delay smaller than the detectable threshold 1ns, and hence are non-detectable. As the most upstream segment opens on the 25.4% branches cause delay smaller than 1ns, all the other opens on their branches cause delay also smaller than 1ns. All opens on the 25.4% are non-detectable and equivalent.

For each branch of the 74.6% branches, if the most upstream segment open causes delay size larger than the tolerated delay threshold 1ns and the most downstream segment open leads to a delay size smaller than 1ns, not all segment opens on this branch are equivalent. Before calculating the delay sizes of the most downstream opens, we filter out Type 3 branches on which open defects at different locations may cause different detection results. 42.7% branch opens cause delay larger than 1μs for all required cell patterns, 1000× detectable threshold 1ns. It is not possible on these branches the delay variation is larger than 1000×. All opens on these branches for all required cell patterns cause delays larger than 1ns, so they are detectable by the same cell patterns and equivalent. For the remaining 31.9% branches, we classify them corresponding to the four types discussed in Section 5, shown in Fig. 21. Only 9.93% (= 2,214) Type 3 branches require the test quality verification.

For all 2,214 Type 3 branches, we calculate the delay size resulting from the most downstream segment open for each pattern. Figure 22 shows per branch, only the minimum delay sizes caused by the most upstream and downstream segment opens by applying all cell patterns. The results

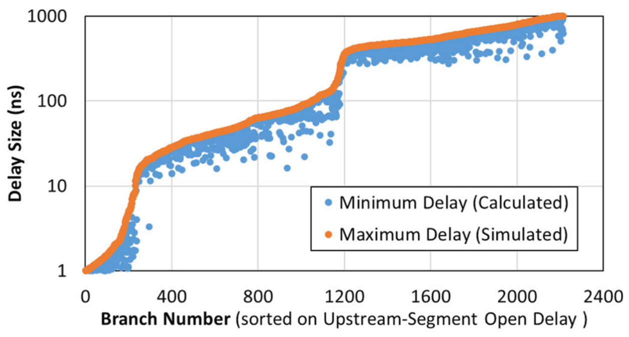


Fig. 22 Minimum delay sizes caused by the-most-upstream and -downstream segment open

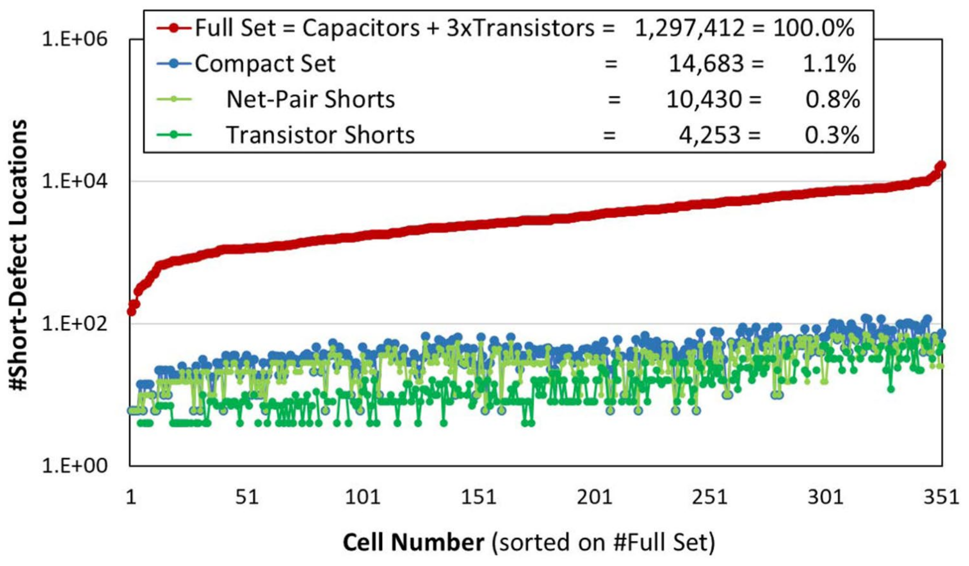


Fig. 23 Full vs.compact sets of short-defect locations in GPDK045 library cells, and the division of the latter into net-pair and transistor shorts

show that on no branch, the most upstream and downstream segment opens cause different detection results. All segment opens on every branch are equivalent. The branch classification and delay size calculation for the Category 5 branches took 0.86 hours in total. In summary, we reduce the open-defect simulation time by 59.0% with maintaining the best test quality.

The PEX tool extracts 1,278,098 parasitic capacitors (excluding self-capacitances). If we assign shorts on all parasitic *C* and four short defects per transistor between its transistor terminals, we identify a full set of $1,278,098 + 3 \times 6,438 = 1,297,412$ short-defect locations. We consider one diagonal short defect per net pair. The library cells together have 22,840 net pairs, for which only 10,430 net pairs have extracted parasitic capacitors. We identify 10,430 net-pair shorts in the compact set. Subsequently, we add additional 4,235 transistor-terminal shorts that are between non-physical interconnects but model transistor-internal defects. The compact set reduces the full set of short locations with $1 - (14,683/(1,278,098 + 3 \times 6,438)) = 98.9\%$. Figure 23 shows per library cell the DLI results for shorts. Unlike what was claimed by [14], there is no need to explicitly add stuck-at faults at the cell boundaries to the candidate defect set, as all stuck-at faults are already included in the compact set as short defects between cell ports and power or ground.

Figure 24 shows per cell the simulation time in the defect characterization and test quality verification and compensation. Assuming we simulate the full set of short defects, for each library cell, the number of simulations equals the product of short-defect count and cell pattern count, resulting in 21,788,604 (=100%) simulations. With the approach proposed in this paper, we first simulate only the compact

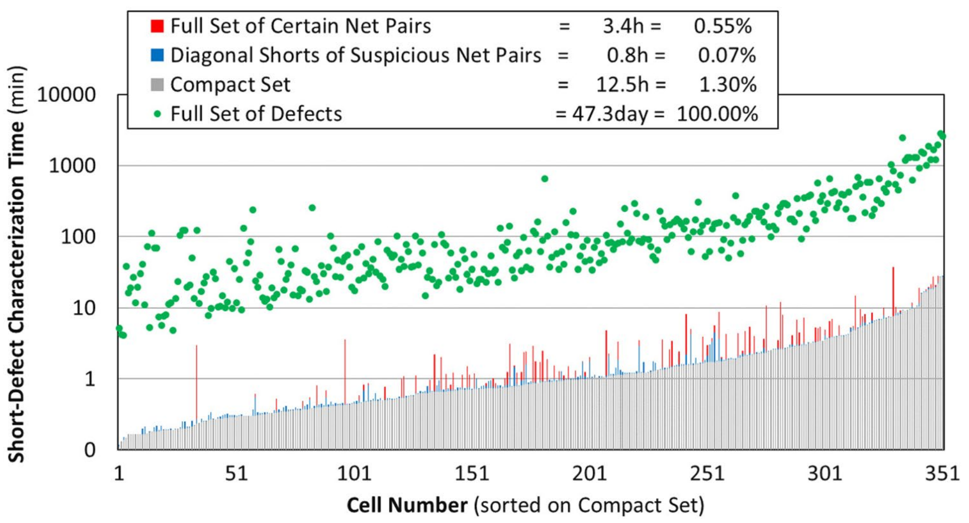


Fig. 24 Simulation times for the short defects in GPDK045 library cells

set with 284,288 (=13%) simulations. The simulation time based on the compact set is 12.5 hours. Via linear extrapolation, the simulation time based on the full set would be 47.3 days, which is clearly unfeasible and underlines the necessity of reducing defect characterization time.

During short-defect characterization, we use the exhaustive set of one-cycle cell patterns to simulate each short defect and record the voltage and current values at each short-defect location for the downstream test quality verification and compensation. Using these values, we identify the combinations of net pairs and cell patterns that may result in different detection results due to different short-defect locations. Between 1,616 (=15.5%) out of 10,430 net pairs, short defects are possible non-equivalent for one or multiple cell patterns. Applying only the cell patterns that can cause different detections, we perform simulations for the other diagonal short defect of the 1,616 net pairs. The total additional number of simulations is 16,232, and the corresponding simulation time is 0.8 hours. After this simulation, we determine 348 (=0.03%) net pairs between which two diagonal short defects are non-equivalent. Therefore, between these net pairs, not all short defects are equivalent. Among the 348 net pairs, 148 net pairs are with only one cell pattern causing different detections for the two diagonal short defects. We already can determine the only two DDM columns caused by these net pair shorts without additional simulation. For the remaining 240 net pairs, we perform additional simulations for the full set of the short defects. 120,115 times simulations took additional 3.4 hours and identified 823 different DDM columns for these net pair shorts. In total, we spent 16.7 hours simulating the short defects; 74.9% simulation is for the compact set of short defects, and 25.1% simulation time is for the test quality verification and compensation. Compared with the estimated simulation time for the full set, we save 98.5% time cost on defect simulation.

8 Conclusion

CAT promises to significantly reduce test-escape rates, as it targets cell-internal defects that are covered by conventional test approaches only on a serendipitous basis. Test quality and costs of CAT critically depend on the library characterization. To guarantee the test quality, we propose a PEX based approach to identify a full set of defect locations. To reduce the simulation time, we minimize the number of defects to be simulated without a negative impact on the test quality. We collapse the full set to a compact set based on the defect equivalence. For opens, this means that we consider only one defect per net branch, while for shorts, we consider only one defect per net pair. Even though simulation with the compact set maintains the test quality for a large number of defects, errors can happen in some specific situations. On particular branches, open-defect location affects the resulting delay size and may lead to different detection results based on the user-defined delay threshold. Between cell-internal net pairs, short-defect location affects the voltage, namely the logic value for the downstream circuit. We propose a solution to verify the test quality based on the compact set and, if necessary, compensate the test quality to the same level based on the full set. In our experiment results for Cadence's GPDK045 library, the defect collapsing reduces the total number of defect locations to be simulated by 96.6% compared to the number of defects in the full set. To compensate the test quality, we simulate additional 2% defects from the full set to identify 823 more different DDM columns. In theory, we would reduce 96.4% simulation time compared with simulating the full set of defects.

Acknowledgments We thank the following persons for their support. At Cadence Design Systems: Maxime Barbe, Vivek Chickermane, Franck Gerome, Walter Hartong, Anton Klotz, Robert McGowan, Michel Montanuy, Sylvie Parmantier, Mike Vachon, Carl Wisnesky, and Vladimir Zivkovic. At IMEC: Kristof Croes, Peter Debacker, Hilde De Witte, Ingrid De Wolf, Arul Mahesh Jagadeesa Das, Alessio Spessot, Ibrahim Tatar, and Diederik Verkest. At TU Delft: Said Hamdioui. At TU Eindhoven: Hailong Jiao. At National Tsing-Hua University: Min-Chun Hu and Cheng-Wen Wu. At University of Brussels: Dragomir Milojevic. Last but not least, we would thank the anonymous reviewer for his insightful question, which inspires us to improve our approach. Zhan Gao is financially supported by the China Scholarship Council (CSC) and Cadence Design Systems. Min-Chun Hu is financially supported by IMEC and National Tsing-Hua University.

References

1. Arumí D, Rodriguez-Montanes R, Figueras J (2008) Experimental Characterization of CMOS Interconnect Open Defects. IEEE Transactions on Computer-Aided Design 27(1):123–136. https://doi.org/10.1109/TCAD.2007.907255
2. Cadence Design Systems (1999) Cadence Standard Parasitic Format (SPF)
3. Cadence Design Systems (2014) Reference Manual Generic 45nm Salicide 1.0V/1.8V 1P 11M Process Design Kit and Rule Decks (PRD) Revision 4.0
4. Chen F et&nbsp;al (2012) Investigation of Emerging Middle-of-Line Poly Gate-to-Diffusion Contact Reliability Issues. In Proc. IEEE International Reliability Physics Symposium (IRPS), pages 6A.4.1–6A.4.9. https://doi.org/10.1109/IRPS.2012.6241865

5. Cho K, Mitra S, McCluskey E (2005) Gate Exhaustive Testing. In Proc. IEEE International Test Conference (ITC), pages 1–7. https://doi.org/10.1109/TEST.2005.1584040
6. Eichenberger S, et&nbsp;al. (2008) Towards a World Without Test Escapes: The Use of Volume Diagnosis to Improve Test Quality. In Proc. IEEE International Test Conference (ITC), pages 1–10. https://doi.org/10.1109/TEST.2008.4700604
7. Eldred R (1959) Test Routines Based on Symbolic Logical Statements. ACM Transactions on Design Automation of Electronic Systems 6(1):33–36. https://doi.org/10.1145/320954.320957
8. Fantini F, Morandi C (1985) Failure modes and mechanisms for VLSI ICs - a review. IEE Proceedings G - Electronic Circuits and Systems 132(3):74–81. https://doi.org/10.1049/ip-g-1.1985.0018
9. Galiay J, Crouzet Y, Vergniault M (1980) Physical Versus Logical Fault Models MOS LSI Circuits: Impact on Their Testability. IEEE Transactions on Computers, C-29(6):527–531. https://doi.org/10.1109/TC.1980.1675614
10. Gao Z et&nbsp;al (2019) Application of Cell-Aware Test on an Advanced 3nm CMOS Technology Library. In Proc. IEEE International Test Conference (ITC), pages 1–6. https://doi.org/10.1109/ITC44170.2019.9000164
11. Gao Z et&nbsp;al (2019) Defect-Location Identification for Cell-Aware Test. In Proc. IEEE Latin-American Test Symposium (LATS), pages 1–6. https://doi.org/10.1109/LATW.2019.8704561. (Best Paper Award)
12. Geuzebroek J, et&nbsp;al. (2007) Embedded Multi-Detect ATPG and Its Effect on the Detection of Unmodeled Defects. In Proc. IEEE International Test Conference (ITC), pages 1–10. https://doi.org/10.1109/TEST.2007.4437649
13. Hapke F et&nbsp;al (2011) Cell-Aware Analysis for Small-Delay Effects and Production Test Results from Different Fault Models. In Proc. IEEE International Test Conference (ITC), pages 1–8. https://doi.org/10.1109/TEST.2011.6139151
14. Hapke F et al (2014) Cell-Aware Test. IEEE Transactions on Computer-Aided Design 33(9):1396–1409. https://doi.org/10.1109/TCAD.2014.2323216
15. Hapke F et&nbsp;al (2009) Defect-Oriented Cell-Aware ATPG and Fault Simulation for Industrial Cell Libraries and Designs. In Proc. IEEE International Test Conference (ITC), pages 1–10. https://doi.org/10.1109/TEST.2009.5355741
16. Hapke F et&nbsp;al (2010) Defect-Oriented Cell-Internal Testing. In Proc. IEEE International Test Conference (ITC), pages 1–10.&nbsp;https://doi.org/10.1109/TEST.2010.5699229
17. Hapke F et&nbsp;al (2011) Gate-Exhaustive and Cell-Aware Pattern Sets for Industrial Designs. In Proc. IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT), pages 1–4. https://doi.org/10.1109/VDAT.2011.5783604
18. Hapke F, Schlöffel J (2012) Introduction to the Defect-Oriented Cell-Aware Test Methodology for Significant Reduction of DPPM Rates. In Proc. IEEE European Test Symposium (ETS), pages 1–6. https://doi.org/10.1109/ETS.2012.6233046
19. Howell W et&nbsp;al (2018) DPPM Reduction Methods and New Defect Oriented Test Methods Applied to Advanced FinFET Technologies. In Proc. IEEE International Test Conference (ITC)
20. Hu M et&nbsp;al (2020) Tightening the Mesh Size of the Cell-Aware ATPG Net for Catching All Detectable Weakest Faults. In Proc. IEEE European Test Symposium (ETS), pages 1–6. https://doi.org/10.1109/ETS48528.2020.9131567
21. Ismail YI, Friedman EG, Neves JL (1999) Figures of Merit to Characterize the Importance of on-Chip Inductance. IEEE Transactions on VLSI Systems 7(4):442–449. https://doi.org/10.1109/92.805751
22. Kaczer B et&nbsp;al (2003) Experimental Verification of SRAM Cell Functionality after Hard and Soft Gate Oxide Breakdowns. In Conference on European Solid-State Device Research, pages 75–78. https://doi.org/10.1109/ESSDERC.2003.1256814
23. Khatri D et&nbsp;al (2016) Simulation Assisted Uncovering and Understanding of Complex Failures in 28nm Microprocessor Devices. In IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA), pages 64–68. https://doi.org/10.1109/IPFA.2016.7564249
24. Liu H, Lin B, Wu C (2016) Layout-Oriented Defect Set Reduction for Fast Circuit Simulation in Cell-Aware Test. In Proc. IEEE Asian Test Symposium (ATS), pages 156–160. https://doi.org/10.1109/ATS.2016.25
25. Ma S, Franco P, McCluskey E (1995) An Experimental Chip to Evaluate Test Techniques Experiment Results. In Proc. IEEE International Test Conference (ITC), pages 663–672. https://doi.org/10.1109/TEST.1995.529895
26. Maxwell P, Hapke F, Tang H (2016) Cell-Aware Diagnosis: Defective Inmates Exposed in Their Cells. In Proc. IEEE European Test Symposium (ETS), pages 1–6. https://doi.org/10.1109/ETS.2016.7519313
27. Majhi AK, Agrawal VD (1998) Tutorial: Delay Fault Models and Coverage. In Proc. 11th International Conference on VLSI Design, pages 364–369. https://doi.org/10.1109/ICVD.1998.646634
28. Pomeranz I, Reddy S (2001) Definitions of the Numbers of Detections of Target Faults and Their Effectiveness in Guiding Test Generation for High Defect Coverage. In Proc. Design, Automation, and Test in Europe (DATE), pages 504–508.&nbsp;https://doi.org/10.1109/DATE.2001.915070
29. Prahov R, Graupner A (2011) A Comprehensive Workflow and Methodology for Parasitic Extraction. In Annual Scientific Conference, Ruse, Bulgaria. http://conf.uni-ruse.bg/bg/docs/cp11/3.1/3.1-23.pdf
30. Rabaey ACJM, Nikolic B (2003)&nbsp;Digital integrated circuits-A design perspective. Prentice Hall, 2003
31. Segura J et&nbsp;al (1995) A Detailed Analysis of GOS Defects in MOS Transistors: Testing Implications at Circuit Level. In Proc. IEEE International Test Conference (ITC), pages 544–551. https://doi.org/10.1109/TEST.1995.529882
32. Shen J, Maly W, Ferguson F (1985) Inductive Fault Analysis of MOS Integrated Circuits. IEEE Design & Test 2(6):13–26. https://doi.org/10.1109/MDT.1985.294793
33. Sinha A et&nbsp;al (2017) DFM-Aware Fault Model and ATPG for Intra-Cell and Inter-Cell Defects. In Proc. IEEE International Test Conference (ITC), pages 1–10. https://doi.org/10.1109/TEST.2017.8242059
34. Spinner S et&nbsp;al (2008) Automatic Test Pattern Generation for Interconnect Open Defects. In Proc. IEEE VLSI Test Symposium (VTS), pages 181–186. https://doi.org/10.1109/VTS.2008.30
35. Stanford EE (2001) Handouts. SPICE Quick Reference Sheet v1.0. https://web.stanford.edu/class/ee133/handouts/general/spice_ref.pdf
36. Tang&nbsp; H et&nbsp;al (2017) Using Cell Aware Diagnostic Patterns to Improve Diagnosis Resolution for Cell Internal Defects. In Proc. IEEE Asian Test Symposium (ATS), pages 231–236. https://doi.org/10.1109/ATS.2017.51
37. Yang F et&nbsp;al (2014) Silicon Evaluation of Cell-Aware ATPG Tests and Small Delay Tests. In Proc. IEEE Asian Test Symposium (ATS), pages 101–106. https://doi.org/10.1109/ATS.2014.29
38. Zhang F, Thornton,M, Dworak J (2014) When Optimized N-Detect Test Sets are Biased: An Investigation of Cell-Aware-Type Faults and N-Detect Stuck-At ATPG. In IEEE North-Atlantic Test Workshop (NATW), pages 32–39. https://doi.org/10.1109/NATW.2014.15

SantoshMalagi is working as a software engineer with Cadence as a part of their Modus ATPG R&D group. His current area of focus is cell-aware test, he has played a significant role in developing a cell-aware test methodology for prospective Cadence customers. He has experience in executing all steps involved in cell-aware test – i.e. Parasitic Extraction, Library Characterization and Cell-Aware ATPG. Santosh recently completed a master's in computer engineering from the Delft University of Technology in the Netherlands, where he got interested in VLSI testing. Previously, he was a research intern on a joint development project on cell-aware test between Cadence and IMEC in Leuven, Belgium.

Joe Swenton earned his B.S. degree in Computer Science from Binghamton University in 1996. He has worked in the EDA field since 1983 for both IBM and Cadence Design Systems, developing solutions for a variety of problems including embedded macro test, stored pattern ATPG, fault analysis and diagnostics. He is presently working in advanced fault modeling of cell internal defects for ATPG and diagnosis.

Jos Huisken is a researcher in the domain of energy efficient VLSI design. After graduation from the University of Twente he joined Philips Research to work on digital signal processor design. He has been involved in architectural synthesis for digital signal processors, and applied these techniques for the first Digital Audio Broadcast (ETSI-DAB) ICs in the 1990s. Since then he has been driving low power VLSI-design from an architectural point of view. After investigating turbo- and LDPC-decoders, and being involved in creating a spinoff company Silicon Hive from Philips on digital signal processing and compilation, he joined Holst Centre/imec in 2008 to work on ultra low power DSP for wireless sensor nodes, specifically for body area networks, with a strong focus on low-voltage and low-power circuit design. Joining RWTH Aachen in 2013 his research shifted to reliable VLSI design and Design for Error Resilience (DfER). On January 2016 he joined TU/e to give courses on VLSI circuit design, carry out research in the field of EEG, ultrasound and baseband processing and to continue his research in Design for Error Resilience.

Kees Goossens is Full Professor of Realtime Embedded Systems in the Electronic Systems group of the Department of Electrical Engineering at Eindhoven University of Technology (TU/e). He focuses on composable (virtualized), predictable (real-time), low-power embedded systems, supporting multiple models of computation. At Topic Embedded Products, Goossens works on real-time dependable dynamic partial reconfiguration in FPGAs. Early in his research, he investigated the formal verification of hardware, in particular by using semi-automated proof systems in conjunction with formal semantics of hardware description languages such as ELLA and VHDL. At NXP Semiconductors, Goossens worked on real-time networks on chip for consumer electronics, then on on-chip communication protocols and memory management. He led the team that defined the Aethereal network on chip for consumer electronics. Goossens has been editorial board member for the ACM Transactions on Design Automation of Electronic Systems (TODAES) since 2009 and associate editor for the Springer Journal of Design Automation of Embedded Systems (DAEM) since 2006, and he has been guest editor for several special issues on networks on chip. He is author of 24 patents, and he has published four books and over 175 articles, with four paper awards.