Paper at IEEE Latin-American Test Symposium (LATS'19) – Santiago, Chile – March 2019

Defect-Location Identification for Cell-Aware Test

Zhan Gao[1,2,3] Santosh Malagi[1,2,4] Erik Jan Marinissen[1,2] Joe Swenton[2] Jos Huisken[3] Kees Goossens[3]

[1] IMEC
Leuven, Belgium
zhan.gao.ext@imec.be
erik.jan.marinissen@imec.be

[2] Cadence Design Systems
Endicott, NY, USA
malagi@cadence.com
jswenton@cadence.com

[3] TU Eindhoven
Eindhoven, the Netherlands
j.a.huisken@tue.nl
k.g.w.goossens@tue.nl

[4] TU Delft
Delft, the Netherlands

Abstract – Cell-aware test (CAT) explicitly targets defects inside library cells and therefore significantly reduces the amount of test escapes compared to conventional automatic test pattern generation (ATPG). Our CAT flow consists of three steps: (1) defect-location identification (DLI), (2) defect characterization based on detailed analog simulation of the cells, and (3) cell-aware automatic test pattern generation (ATPG). This paper focuses on Step 1, as quality and cost are determined by the set of cell-internal defect locations considered in the remainder of the flow. Based on technology inputs from the user and a parasitic extraction (PEX) run that analyzes the cell layouts, we derive a set of open defects on and short defects between both transistor terminals and intra-cell interconnects. The full set of defect locations is stored for later use during failure analysis. Through dedicated DLI algorithms, we identify a compact subset of defect locations for defect characterization and ATPG, in which we include only one representative defect location for each set of equivalent defects locations. For Cadence's GPDK045 library, the compact subset contains only 2.8% of the full set of defect locations and reduces the time required for defect characterization with the same ratio.

Keywords – cell-aware test, defects, open, short, parasitic extraction, defect detection matrix, equivalence

1 Introduction

Advanced semiconductor technology nodes have tiny feature sizes, complex transistor architectures, and a large number of interconnect layers. This allows integration of incredibly many devices into a single IC, but also makes these 'monster chips' sensitive to manufacturing defects. Increasingly, such ICs are used in safety-critical applications, such as automotive and healthcare, where defects have profound consequences and test escapes cannot be tolerated. Consequently, there is an industry-wide need for test quality improvement.

Digital logic ICs are designed on the basis of a library of standard cells. Conventional ATPG tools target faults on the interconnects between cells, such as stuck-at and transition/delay faults. Intra-cell defects are typically not on the radar screen of conventional ATPG tools, and hence only detected fortuitously. Not surprisingly, cell-internal defects are found to cause a significant fraction of test escapes [1]. Cell-aware test (CAT) is a three-step test generation approach that explicitly targets cell-internal defects. In Step 1, *defect-location identification*, we determine for each library cell on the basis of its layout the set of locations where cell-internal defects might occur. In Step 2, *defect characterization*, we select the defect size for these locations and determine through analog simulation which cell-level test patterns detect which cell-internal defects; the result is encoded in a *defect detection matrix* (DDM). Step 3, *cell-aware ATPG*, uses cell-patterns as starting point for chip-level test pattern generation. It tries to maximize the coverage of the detectable cell-internal defects for the DDM of the cell type, while minimizing the number of chip patterns.

In this paper, we focus on DLI, as that step is fundamental to both test quality and cost of CAT. The set of locations for potential defects should be realistic and complete. Too few defect locations means the test might be missing defects; this can negatively impact the test and product quality. Too many defect locations imply unnecessary time-consuming analog simulations during defect characterization. We automatically identify potential open defects on and short defects between both transistor terminals and intra-cell interconnects. We use PEX not only in its conventional role to create accurate analog simulation models of library cell layouts, but also to analyze the cells' layouts to identify possible defect locations on interconnects. The number of identified defect locations quickly grows large, even for small library cells. Many of them are considered equivalent, as their stimulus vectors and resulting fault effects are identical. Therefore, we employ dedicated DLI algorithms to determine a compact set with defect locations, in which we include only one representative defect location per set of equivalent defect locations. The compact set is used for subsequent analog defect simulation and CA-ATPG. This reduces the compute time significantly, simply because the compact set is significantly smaller. The full defect-locations set is stored for future diagnosis purposes during failure analysis. This approach has been integrated in the Cadence CAT flow.

The remainder of this paper is organized as follows. Section 2 reviews related prior work. Cadence's CAT flow is introduced in Section 3. The PEX settings for this flow are described in Section 4, while the post-PEX DLI algorithms for identifying per equivalence set one open and one short defect, are described in Sections 5 and 6, respectively. Section 7 presents experimental results and Section 8 concludes this paper.

2 Related Prior Work

Recently, CAT attracts a lot of attention in the IC industry. When introduced by Hapke et al. in 2009 [2], initially, only hard short defects (1Ω) in combinational logic cells were targeted. Later, this was extended to hard open defects ($1G\Omega$), weaker short defects (1Ω – $20k\Omega$) [3, 4], and sequential cells such as scan flip-flops [5]. Several studies have shown that CAT offers superior test quality and reduces test cost in comparison with n-detect [6–8], embedded multi-detect [9, 10], and gate-exhaustive test [11, 12]. Industrial application has provided experimental evidence with respect to the effectiveness of CAT to reduce test escapes [4, 8, 13–15]. CAT also improves the diagnosis accuracy and efficiency by recording the layout location of potential defects [16, 17].

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 ac-

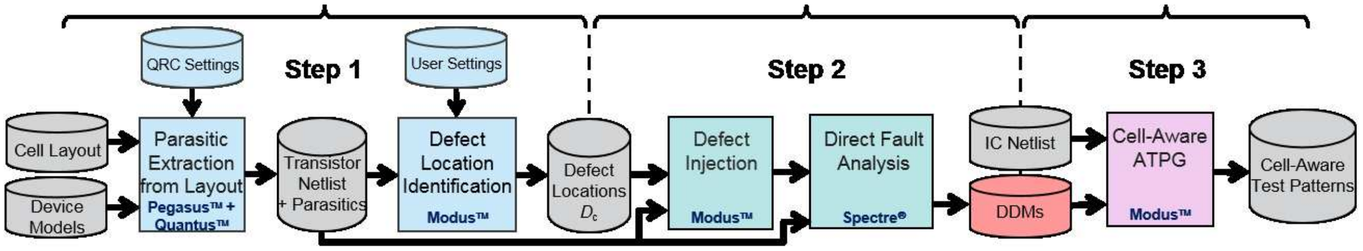
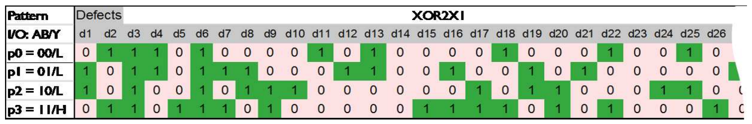


Figure 1: Cadence's CAT tool flow consists of three main steps: (1) defect-location identification, (2) defect characterization, and (3) cell-aware ATPG.

tually contribute to the detection of these defects. Prior work does not disclose 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; in this paper, we do. Other papers claim to use PEX for DLI, but do not report how the many user controls of a PEX tool were set [2, 18, 19]; we do. Prior publications present different opinions regarding the inclusion of inter-layer shorts: Hapke et al. include them [2, 19], while Liu et al. claim they do not occur and hence can be omitted [20]. We support both options, as we have observed that in advanced-node technologies, inter-layer shorts can indeed occur, 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, [19] claims to include also shorts, opens, stuck-at, and transition faults at the cell I/Os in cell-aware ATPG; we demonstrate that these cell-boundary defects are often equivalent to already modeled cell-internal defects.

3 Cadence's Cell-Aware Test Tool Flow

The Cadence CAT tool flow is depicted in Figure 1. It generates as output of Step 2 and input for Step 3 a DDM per library cell. A DDM records which cell-patterns detect which cell-internal defects. The DDM for an example two-input XOR cell is partially shown in Figure 2. A DDM is a binary matrix. Its rows correspond to *single-output cell-patterns* which contain stimulus bits for one or more cell inputs and the corresponding defect-free response bit on a single cell output. The columns correspond to defects that have been identified to potentially occur in the cell. DDM entry (p, d) denotes whether cell-pattern p detects defect d (denoted by '1') or not (denoted by '0').



Pattern	Defects												XOR2X1													
I/O: 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
p0 = 00/L	0	1	1	1	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0
p1 = 01/L	1	0	1	1	0	1	1	1	0	0	0	1	1	0	0	1	0	0	1	0	1	0	0	0	0	0
p2 = 10/L	1	0	1	0	0	1	0	1	1	1	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1	0
p3 = 11/H	0	1	1	0	1	1	1	0	1	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	0	1

Figure 2: Example (partial) defect detection matrix for a two-input AND cell.

For all cells c in library C, in Step 1, Cadence's Pegasus™ layout-versus-schematic (LVS) generates files required by PEX. Subsequently PEX is performed by Cadence's Quantus™ Extraction Solution. On the basis of a cell's transistor netlist with extracted parasitic resistors and capacitors, the DLI function in Modus™ determines for each cell $c \in C$ a set D_c with possible open- and short-defect locations.

In Step 2, Modus' defect characterization function adds a resistive value to the defect locations in D_c; this allows us to model hard as well as weak resistive open and short defects. The defects and the defect-free netlist are submitted to the *Direct Fault Analysis* function of Cadence's analog simulator Spectre® [21]. It generates $|D_c| + 1$ netlists: $|D_c|$ defective netlists with one defect from D_c per netlist, and one defect-free netlist. These are simulated for the exhaustive set P_c of single-output cell-patterns, with $|P_c| = m \times 2^n$ for a cell c with n inputs and m outputs. Only if the simulation of cell-pattern $p \in P_c$ for the cell with defect $d \in D_c$ gives a response which differs from the defect-free cell on the involved single-output of the cell, we mark in the DDM for c the entry for defect d and pattern p as 'detected'. For *all* library cells $c \in C$, Spectre simulates *all* defects $d \in D_c$ with *all* possible single-output cell-patterns $p \in P_c$: this is time consuming, especially for larger library cells with many inputs. Fortunately, this task needs to be executed once per library release, while the resulting DDMs can be reused for all IC designs based on the same library.

Step 3 is cell-aware ATPG. It uses as inputs the IC's netlist with library cells as its atomic building blocks, and the set of DDMs for these library cells. The cell-patterns at the cell instances are translated into ATPG targets, which Modus' ATPG engine' tries to expand into chip-level test patterns. If expansion is successful, the cell-internal defects which are, according to the DDM, detected by the cell-pattern, count as covered. The objective of cell-aware ATPG is to cover as many as possible cell-internal defects in the cell instances with a minimum set of chip-patterns.

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. A PEX tool analyzes a circuit using the circuit layout and a set of technology parameters (such as the per-layer sheet resistance of interconnects) as inputs. 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 [22]. In standard cells, intra-cell distances and hence L values are very small and therefore can be ignored.

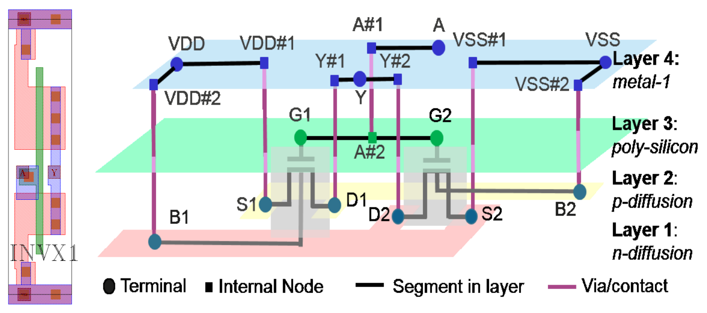
We use Cadence Quantus to perform PEX on the cell. Some of the general settings for Quantus are specified in Figure 3. A PEX tool divides a *net* that electrically connects two or more *terminals* into net *segments*, which are bounded by terminals and/or *internal nodes*. Reasons for a PEX tool to split a net into multiple separate segments can be (1) a *fork* in a net with multiple destinations, (2) vertical interconnects from one layer to another (referred to as *via* or *contact*), or (3) a 90° bend within a layer. Using complex sets of equations and advanced field solvers, the PEX tool calculates a parasitic R for every net segment, and a parasitic C between every pair of nodes [23].

We illustrate the above model with a cell from the Cadence GPDK045 library [24] as example. This cell library utilizes four layers: n-diffusion, p-diffusion, poly-silicon, and metal-1; contacts exist from metal-1 to any of the three other layers. Figure 4(a) shows the layout

- `input_db -type pvs`
  Set input format to the database format as generated by Cadence Pegasus.
- `extract -type rc_coupled`
  As library cells are per definition small, we need to extract only resistor (R) and capacitor (C) values, while inductors (L) can safely be ignored.
- `output_db -add_bulk_terminal true`
  `output_db -add_explicit_vias true`
  `output_db -em_extract true`
  These three options specify that defect-susceptible circuit elements are included in the PEX output netlist: (1) transistor bulk terminals, vias (which in library cells are typically called "contacts"), and 90° bends ('L shapes') in nets (identified by EM analysis).
- `output_db -type dspf`
  We use DSPF (Cadence's detailed standard parasitic format) as output format. The benefits in the CAT context of DSPF over for example Spice format are (1) an explicit layer list, (2) explicit listing of nets, and (3) explicit listing of net segments in diffusion layers.
- `output_db -include_parasitic_res_model "comment"`
  `output_db -include_parasitic_res_width true`
  `output_db -output_xy "PARASITIC_RES" "PARASITIC_CAP"`
  Include as comments into the netlist for usage during diagnosis, for all parasitic R's and C's: layer information, x,y location coordinates, and segment widths.

Figure 3: Quantus general settings for PEX in CAT flow.

of inverter cell INVX1 and Figure 4(b) the corresponding cell model. Cell INVX1 has one input (A), one output (Y), and two transistors; consequently, there are twelve terminals. In addition, the PEX tool has identified eight internal nodes. The total set of 20 nodes is partitioned over four nets: $Nets_{\text{INVX1}} = \{\{A, A\#1, A\#2, G1, G2\}, \{Y, Y\#1, Y\#2, D1, D2\}, \{V_{DD}, V_{DD}\#1, V_{DD}\#2, S1, B1\}, \{V_{SS}, V_{SS}\#1, V_{SS}\#2, S2, B2\}\}$.



(a) (b)

Figure 4: Example library cell INVX1: (a) layout and (b) PEX cell model.

5 Identification of Open-Defect Locations

Our CAT tool flow identifies open-defect locations on and short-defect locations between both transistor terminals and intra-cell interconnects. In this section, we deal with open defects; in the next section we address shorts.

We analyze for each library cell its layout by letting the PEX tool extract all relevant net segments and their parasitic resistor values; the associated user settings for Quantus are listed in Figure 5. A segment's parasitic resistor value is indicative of the probability that that segment will suffer from an open defect: a large R value indicates that the corresponding interconnect is thin and/or long and both increase the probability for that interconnect to be affected by an open defect. To allow comparison of R values of segments in different layers, we normalize them by division by the sheet resistance of the interconnect layer.

Subsequently, Modus' DLI function identifies a compact set open-defect locations on transistor terminals and on selected net segments. For an electrical test, open defects on concatenated segments of a single net are equivalent with respect to their test stimuli and expected test responses. Only if a net forks out to multiple destinations, opens on the different *branches* of the fork can be distinguished and hence will require different test patterns. To minimize the number of open-defect locations and associated Spectre simulation times during downstream defect characterization, the DLI function identifies at most one open defect location per net branch. However, for diagnostic purposes, we store *all* eligible segments, information about the individual segments per branch, including their layout location and processing layer.

- `extraction_setup -max_fracture_length infinite`
  With this setting, we instruct Quantus *not* to break segments up in multiple segments only because the length of the segment exceeds a limit.
- `filter_res -merge_parallel_res true`
  Parallel segments (other than parallel vias and contacts) are expected not to occur in the heavily-optimized circuits that standard-cell libraries are.
- `filter_res -merge_parallel_via`
  `filter_res -max_via_array_size "auto"`
  The only type of parallel R segments that occur frequently in library cells are parallel vias. Multiple parallel vias are meant to increase the yield, and effectively work as one (lower-resistance) parallel connection. This setting merges 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 to be defective.
- `filter_res -remove_dangling_res true`
  This option suppresses listing of parasitic resistors in dangling segments, while any capacitors to their nodes remain. As the missing R's do not affect the simulation accuracy and open defects in dangling segments would not cause faulty behavior, we utilize this option.
- `filter_res -min_res 1e-10`
  This option filters out parasitic resistors below a specified limit (in Ohm). We want the PEX tool to include *all* resistor segments in the cell model, and optionally apply our own R filtering in the DLI function. Quantus does not accept a zero limit, therefore we specify as limit 10^{-10} Ω.

Figure 5: Quantus settings for identification of potential open defects.

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

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

Algorithm 1 [OPEN-DEFECT-LOCATION IDENTIFICATION]

```
01: for all $c \in Cells$ do {
02:   for all $b \in Branches_c$ do { $branchOpen[b] = 0$; }
03:   if ¬ disableTrTerminalOpens then {
04:     for all transistors $t$ in $c$ do {
05:       for all $tm \in TrTerminalOpenSet$ do {
06:         if $t.tm$ is a fork node do {add open defect on transistor terminal $t.tm$;}
07:         else { if $branchOpen[branch(t.tm)] = 0$ then {
08:           add open defect on transistor terminal $t.tm$;
09:           $branchOpen[branch(t.tm)] = 1$;
10:   } } } } };
11:   for all $b \in Branches_c$ do {
12:     if $branchOpen[b] = 0$ then {
13:       $firstOpenFound$ = false;
14:       for all $segm(i,j) \in b$ do {
15:         if $R(i,j) \geq R_{\text{th}}(layer(i,j))$ then {
16:           if $(\neg firstOpenFound) \vee (R(i,j)/R_{\text{sheet}}(layer(i,j)) >$
17:                               $R(mi,mj)/R_{\text{sheet}}(layer(mi,mj)))$ then {
18:             $firstOpenFound$ = true;
19:             $mi = i; mj = j$;
20:       } } };
21:       if $firstOpenFound$ then {
22:         add open defect on segment $(mi,mj)$;
23:         $branchOpen[branch(mi,mj)] = 1$;
24: } } } }
```

To prevent the modeling of multiple open defects on the same branch, Algorithm 1 stores in array *branchOpens[]* with binary values which branches already have been assigned an open defect. In Line 02, this array is initialized to 0. Subsequently (Lines 03–10), for all transistors open locations on the terminals of *TrTerminalOpenSet* are identified, if not disabled by the user (Line 03). In case a transistor terminal is a fork node, it becomes a transistor-open location that is not equivalent to any branch-open defect. Next, open defects on the cell-internal nets are identified (Lines 11-24) per branch and per segment. We require that the branch does not already contain another open defect (Lines 07 and 12) and that the parasitic resistor value of the segment exceeds the user-defined threshold $R_{\rm th}(\ell)$ (Line 15). If there are multiple eligible segments on a given branch, we select the segment which has the largest number of squares (i.e., its parasitic resistance divided by the sheet resistance for that layer), as that indicates the most likely location for an open defect (Lines 16–17).

6 Identification of Short-Defect Locations

This section describes how we handle defect-location identification for short defects. We let the PEX tool extract the parasitic coupling capacitors between all relevant nodes in the library cell; the associated user settings for Quantus are listed in Figure 6. The parasitic capacitor between a pair of nodes belonging to different nets 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 defects both within a layer as well as in between physically adjacent layers.

- `filter_cap -exclude_self_cap False`
  All self-coupling capacitors between nodes on the same net are extracted to obtain an accurate simulation model. The DLI script will ignore these self-coupling capacitors for short defect identification, as we do not want to consider short defects on a single net.
- `filter_cap -exclude_floating_nets true`
  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, this command removes them. A short connected to floating nets cannot cause a fault.
- `filter_coupling_cap`
  `-coupling_cap_threshold_absolute 1e-25`
  `-coupling_cap_threshold_relative 1e-3`
  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 to pose a realistic risk for short defects. 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 the limit at the (very low) 10^{-25} Farad.

Figure 6: Quantus settings for identification of potential short defects.

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

- Parameter *disableTrTerminalShorts* allows to explicitly disable the identification of short defects 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 *blacklist* specifies which inter-layer 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 [25] (default: empty list).
- Threshold value $C_{\rm th}$ defines that extracted parasitic capacitors will not be identified as short defects if $C < C_{\rm th}$ (default: $C_{\rm th} = 0$).

Note that the number of node pairs grows quadratically in the number of nodes. Even if we exclude the pairs of nodes on the same net (corresponding to so-called *self-capacitances* in the PEX tool), which would lead to irrelevant shorts of a net with itself, the number of node pairs of nodes from different nets grows very rapidly with the total number of nodes. However, also note that if nodes of disjoint nets are indeed shorted, their entire nets are shorted and additional shorts between other nodes of the same two nets only lead to equivalent faults. To minimize the number of short defects and associated simulation times during downstream defect characterization, the DLI function identifies at most one short defect per net pair.

Algorithm 2 [SHORT-DEFECT-LOCATION IDENTIFICATION]

```
01: for all $c \in Cells$ do {
02:   for all $n_1, n_2 \in Nets_c$ with $n_1 \neq n_2$ do { $netShort[n_1, n_2]$ = 0; };
03:   if $\neg$ disableTrTerminalShorts then {
04:     for all transistors $t$ in $c$ do {
05:       for all transistor terminal pairs $(tm_1, tm_2) \in TrTerminalShortSet$ do {
06:         if $netShort[net(tm_1),(net(tm_2)]$ = 0 then {
07:           add short defect for transistor terminal pair $tm_1 \leftrightarrow tm_2$;
08:           $netShort[net(tm_1),net(tm_2)]$ = 1; $netShort[net(tm_2),net(tm_1)]$ = 1;
09:   } } } };
10:   for all $n_1, n_2 \in Nets_c$ with $n_1 \neq n_2$ do {
11:     if $netShort[n_1, n_2]$= 0 then {
12:       firstShortFound = false;
13:       for all $i \in n_1$ and $j \in n_2$ do {
14:         if $((layer(i, j)) \notin BlackList) \wedge (C(i, j) \geq C_{\rm th})$ then {
15:           if $(\neg firstShortFound) \vee (C(i, j) > C(mi, mj))$ then {
16:             firstShortFound = true;
17:             $mi = i; mj = j;$
18:       } } };
19:       if firstShortFound then {
20:         add short defect for node pair $(mi, mj)$;
21:         $netShort[n_1, n_2]$ = 1; $netShort[n_2, n_1]$ = 1;
22: } } } }
```

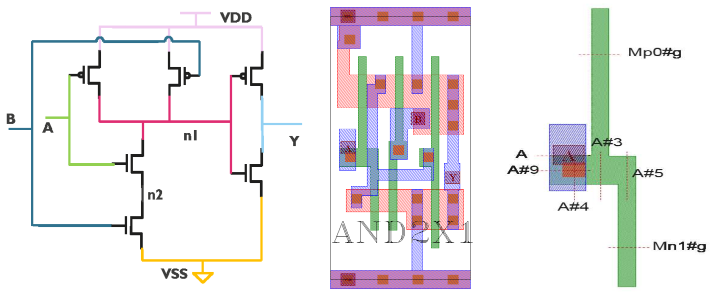
The DLI operation to identify short-defect locations within a library cell is described in Algorithm 2. As we want to identify at most one short per net pair, we use array *netShorts*[] to store which net pairs already have been assigned a short defect; in Line 02, this array is initialized. Subsequently, for all transistors short-defect locations between the terminals of *TrTerminalShortSet* are identified, only if not disabled by the user (Line 03) and there is not already a short between the two corresponding nets (Line 06). Next, the parasitic Cs are evaluated to identify locations for short defects (Lines 10–22) between intra-cell nets. For each net pair $n_1, n_2 \in Nets_c$ which does not have a short identified yet (Line 11), we select the node pair $i \in n_1$ and $j \in n_2$ with the largest $C(i, j)$ that is not on the blacklist and for which holds $C(i, j) \geq C_{\rm th}$ (Line 14).

7 Experimental Results

The experiments reported in this paper are with Cadence's GPDK045 45nm CMOS library [24], which contains 351 combinational and 197 sequential standard cells. The EDA tool versions used are Cadence Pegasus v15.2, Quantus Extraction Solution v18.1, and Modus v19.1. The user settings of the DLI algorithms are mostly defaults; the only exceptions are (1) $R_{\rm th}$ for the diffusion layers, which we specified high enough to exclude open defects in these layers, and (2) $C_{\rm th} = 10^{-20}$ F.

First, we illustrate our approach with cell AND2X1. This 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 29 terminals for this cell. As shown in Figure 7(a), these terminals are interconnected by seven nets: $\{A, B, Y, V_{\rm DD}, V_{\rm SS},$

n1, *n2*}.



(a) Schematic. (b) Layout. (c) Nodes on Net A.

Figure 7: GPDK045 library cell AND2X1.

Figure 7(b), shows the layout of AND2X1. The PEX tool partitions the seven nets into 51 segments, each with its own parasitic resistor. Figure 7(c) illustrates the PEX partitioning of nets into segments for example 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 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 $\{A\#3, A\#5, Mn1\#g\}$. Opens on the latter two branches are equivalent to already identified transistor-terminal opens. In this example, we only need to identify an additional open-defect location on the first branch; the contact is found to be the most likely location.

For the complete cell AND2X1, identifying open defects on *all* gate, source, and drain transistor terminals and on *all* segments would have resulted in 18+51 = 69 open-defect locations. However, AND2X1 has 28 branches and we identify branch opens. Four branches are single-segment bulk-to-bulk nets in diffusion layers, for which no open defects are considered. Three additional transistor-terminal opens are identified as these terminals are fork nodes. So, only 27 out of 69 (= 39.1%) open-defect locations are selected. Hapke et al. [19] include also *port-opens* at the cell boundary in their cell-aware defect set. However, two of the three port-opens for cell AND2X1 are identical to cell-internal open defects and therefore redundant.

Next, we consider short defects. 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 734 parasitic capacitors above its user-specified PEX minimum of 10^{-25} F. Subsequently, the DLI function also has a threshold, C_{th}. In our experiments, we used as C_{th} the minimum extracted parasitic C in the library cell with minimum cell width, The motivation is that in wider cells, many nets are further apart than in the minimum-width cell and hence are unlikely to be shorted. In the GPDK045 library, the cell with minimum width is inverter INVX1, and its minimum capacitor is $C_{\text{th}} = 10^{-20}$ F. Applying threshold C_{th} leaves only 606 of the extracted 734 capacitors as potential short defects. The DLI function considers at most one short defect per net pair. Cell AND2X1 has seven nets, and hence there are $\binom{7}{2}$ = 21 net pairs for consideration. The DLI function first identifies short defects between transistor terminals. Among the 24 transistor-terminal shorts (four per transistor), nine defects are equivalent. Therefore only 24-9 = 15 transistor-terminal short defects are identified. Now only 21-15 = 6 net pairs are left without an already-identified transistor-terminal short between them. The parasitic capacitors of net pairs *net2*-V_{DD} and *net2*-Y are smaller than 10^{-25} F and hence not extracted. The other four net pairs each have capacitors between them that exceed C_{th}, and hence are added to the list of short-defect locations. Three of these shorts (A-Y, B-Y, and A-B) are equivalent to the *port-shorts* at the cell boundary, and hence there is no need to add additional port-shorts to the set of defect locations [19]. In total, 19 of the 21 potential net-pair shorts are considered realistic in our approach; this constitutes a reduction of 97.5% from the 734+24 = 758 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 19 identified cell-internal short defects also cover the six stuck-at faults at the cell boundary, i.e., Ports A, B, and Y either stuck-at-zero or -one, and hence explicit modeling of these stuck-at faults in addition to the cell-aware faults [19] is *not* required.

The PEX run on all 548 cells of the GPDK045 library [24] resulted in 14,730 transistors, 127,503 extracted net segments with parasitic resistors, and 3,757,359 extracted parasitic capacitors, and took 24.2 hours. Note that there are 29.5× more Cs extracted than Rs.

Assigning open defects to three terminals for each transistor and to all net segments would have resulted in 3 × 14,730 + 127,503 = 171,720 open defects. The number of open defects identified by Algorithm 1 equals the number of net branches minus bulk-to-bulk diffusion branches plus the transistor-terminal fork nodes: 75,557 - 13,509 + 9,238 = 71,286. This constitutes a reduction of 58.5%. 41,541 (= 58.3%) of those are transistor-terminal opens and 29,745 (= 41.7%) branch opens. The DLI results per library cell for opens are shown in Figure 8. The compact set with open-defect locations identified by our approach covers 1,831 (= 66.0%) of the total-possible 2,775 port opens [19].

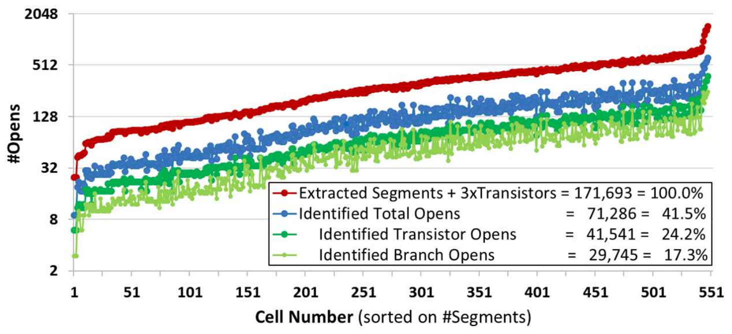


Figure 8: The full vs. the identified (= compact) set of open defects and the division of the latter into transistor and branch opens.

The 548 library cells have a total sum of 21,426,346 node pairs on disjoint nets. The PEX tool only extracts 3,757,359 parasitic capacitors (excluding self-capacitances), of which 2,340,826 are larger than $C_{\text{th}} = 10^{-20}$ F. If we would assign four short defects per transistor between its transistor terminals and shorts on all parasitic $C > C_{\text{th}}$, we would identify 4 × 14,730 + 2,340,826 = 2,399,746 short defects. However, we consider at most one short defect per net pair. The library cells together have 93,056 net pairs, for which we identify 41,882 short defects out of which 26,996 (= 0.7%) shorts are between transistor terminals. The compact set reduces the full set of short locations with

1 - (41,882/(4 $\times$ 14,730 + 3,757,359)) = 98.9%. Figure 9 shows per library cell the DLI results for shorts. Unlike claimed by [19], there is no need to explicitly add stuck-at faults at the cell boundaries to the fault model, as all cell-boundary faults are already included in the cell-aware fault model as short defects between cell ports and power or ground.

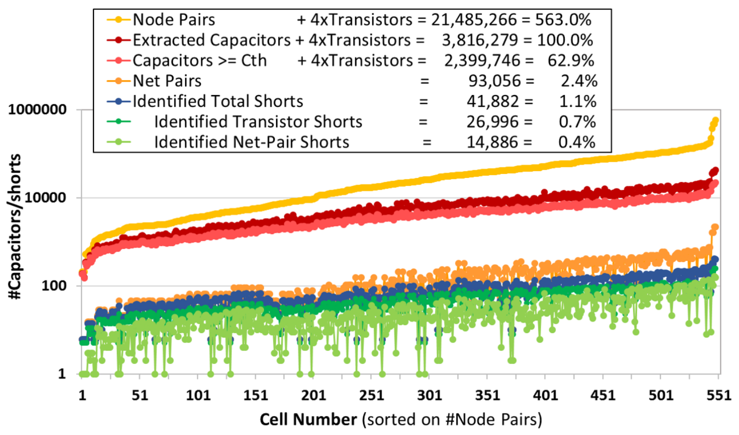


Figure 9: The theoretical number of node pairs, the full, the C_{th}-filtered, vs. the identified (= compact) set of short defects, and the division of the latter into transistor and net-pair shorts.

Table 1 presents the number of identified open and short defect locations in both full and compact sets for all GPDK045 library cells, split into combinational and sequential cells. Defect characterization through Spectre simulations of the compact set for only the combinational cells takes 8.6 hours compute time. This set comprises only 1.18% of the full defect set for the entire library. Hence, defect characterization of the full defect set would, via extrapolation, take $\sim$30 days. This is clearly infeasible and underlines the necessity of our DLI algorithms.

	Defects	Combinational: 351	Sequential: 197	All Cells: 548	C/F
Full	Opens	72,281 (= 5.3%)	99,412 (= 3.8%)	171,693 (= 4.3%)	100%
	Shorts	1,303,850 (= 94.7%)	2,512,429 (= 96.2%)	3,816,279 (= 95.7%)	100%
	Total	1,376,131 (= 100%)	2,611,841 (= 100%)	3,987,972 (= 100%)	100%
Compact	Opens	31,988 (= 67.4%)	39,298 (= 69.8%)	71,286 (= 63.0%)	41.5%
	Shorts	15,438 (= 32.6%)	26,444 (= 40.2%)	41,882 (= 37.0%)	1.1%
	Total	47,426 (= 100%)	65,742 (= 100%)	113,168 (= 100%)	2.8%

Table 1: Identified open and short defect locations for the GPDK045 library.

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 identifying a complete set of realistic defect locations. The DLI approach presented in this paper identifies both open-defects on and short-defects between transistor terminals as well as intra-cell interconnects. For the latter, we use extracted parasitic Rs and Cs as an indicator where defects realistically might occur. The paper presents the settings we use for our PEX tool to obtain both an accurate analog simulation model, as well as useful information for our DLI algorithms to identify all realistic defect locations. Among defect locations with equivalent fault behavior, we model only one defect for downstream library characterization and ATPG, in order to limit the associated costs. For opens, this means that we consider only one defect per net branch, while for shorts we consider only one defect per net pair. In our experimental results for the Cadence GPDK045 library, this reduces the number of open and short defects to be simulated with resp. 58.5% and 98.9% compared to all possible defect locations. To facilitate future diagnosis and failure analysis, we do store key data on *all* equivalent defect sites.

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. Zhan Gao is financially supported by the China Scholarship Council (CSC) and Cadence Design Systems.

References

[1] S. Eichenberger et al. 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, October 2008. doi:10.1109/TEST.2008.4700604.
[2] F. Hapke et al. Defect-Oriented Cell-Aware ATPG and Fault Simulation for Industrial Cell Libraries and Eesigns. In *Proc. IEEE International Test Conference (ITC)*, pages 1–10, November 2009. doi:10.1109/TEST.2009.5355741.
[3] F. Hapke et al. Defect-Oriented Cell-Internal Testing. In *Proc. IEEE International Test Conference (ITC)*, pages 1–10, November 2010. doi:10.1109/TEST.2010.5699229.
[4] F. Hapke et al. 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, September 2011. doi:10.1109/TEST.2011.6139151.
[5] A. Touati et al. Scan-Chain Intra-Cell Aware Testing. *IEEE Transactions on Emerging Topics in Computing*, 6(2):278–287, April 2018. doi:10.1109/TETC.2016.2624311.
[6] S.C. Ma, P. Franco, and E.J. McCluskey. An Experimental Chip to Evaluate Test Techniques Experiment Results. In *Proc. IEEE International Test Conference (ITC)*, pages 663–672, October 1995. doi:10.1109/TEST.1995.529895.
[7] I. Pomeranz and S.M. Reddy. 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, March 2001. doi:10.1109/DATE.2001.915070.
[8] F. Hapke and J. Schlöffel. 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, May 2012. doi:10.1109/ETS.2012.6233046.
[9] J. Geuzebroek et al. Embedded Multi-Detect ATPG and Its Effect on the Detection of Unmodeled Defects. In *Proc. IEEE International Test Conference (ITC)*, pages 1–10, October 2007. doi:10.1109/TEST.2007.4437649.
[10] F. Zhang, M. Thornton, and J. Dworak. 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, May 2014. doi:10.1109/NATW.2014.15.
[11] K.Y. Cho, S. Mitra, and E.J. McCluskey. Gate Exhaustive Testing. In *Proc. IEEE International Test Conference (ITC)*, pages 1–7, November 2005. doi:10.1109/TEST.2005.1584040.
[12] F. Hapke et al. 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, April 2011. doi:10.1109/VDAT.2011.5783604.
[13] F. Yang et al. Silicon Evaluation of Cell-Aware ATPG Tests and Small Delay Tests. In *Proc. IEEE Asian Test Symposium (ATS)*, pages 101–106, November 2014. doi:10.1109/ATS.2014.29.
[14] A. Sinha et al. DFM-Aware Fault Model and ATPG for Intra-Cell and Inter-Cell Defects. In *Proc. IEEE International Test Conference (ITC)*, pages 1–10, October 2017. doi:10.1109/TEST.2017.8242059.
[15] W. Howell et al. DPPM Reduction Methods and New Defect Oriented Test Methods Applied to Advanced FinFET Technologies. In *Proc. IEEE International Test Conference (ITC)*, October 2018.
[16] P. Maxwell, F. Hapke, and H. Tang. Cell-Aware Diagnosis: Defective Inmates Exposed in Their Cells. In *Proc. IEEE European Test Symposium (ETS)*, pages 1–6, May 2016. doi:10.1109/ETS.2016.7519313.
[17] H. Tang et al. Using Cell Aware Diagnostic Patterns to Improve Diagnosis Resolution for Cell Internal Defects. In *Proc. IEEE Asian Test Symposium (ATS)*, pages 231–236, November 2017. doi:10.1109/ATS.2017.51.
[18] S. Spinner et al. Automatic Test Pattern Generation for Interconnect Open Defects. In *Proc. IEEE VLSI Test Symposium (VTS)*, pages 181–186, April 2008. doi:10.1109/VTS.2008.30.
[19] F. Hapke et al. Cell-Aware Test. *IEEE Transactions on Computer-Aided Design*, 33(9):1396–1409, September 2014. doi:10.1109/TCAD.2014.2323216.
[20] H.W. Liu, B.Y. Lin, and C.W. Wu. Layout-Oriented Defect Set Reduction for Fast Circuit Simulation in Cell-Aware Test. In *Proc. IEEE Asian Test Symposium (ATS)*, pages 156–160, November 2016. doi:10.1109/ATS.2016.25.
[21] Art Schaldenbrand and others. Improving Test Coverage and Eliminating Test Escapes Using Analog Defect Analysis. `https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/custom-ic-analog-rf-design/legato-reliability-wp.pdf`.
[22] R. Prahov and A. Graupner. A Comprehensive Workflow and Methodology for Parasitic Extraction. In *Annual Scientific Conference, Ruse, Bulgaria*, November 2011. `http://conf.uni-ruse.bg/bg/docs/cp11/3.1/3.1-23.pdf`.
[23] X. Qi and R.W. Dutton. Interconnect Parasitic Extraction of Resistance, Capacitance, and Inductance. In Jeffrey A. Davis and James D. Meindl, editor, *Interconnect Technology and Design for Gigascale Integration*, chapter 3, pages 67–109. Kluwer Academic Publishers, 2003. doi:10.1007/978-1-4615-0461-0.
[24] Cadence Design Systems. Reference Manual Generic 45nm Salicide 1.0V/1.8V 1P 11M Process Design Kit and Rule Decks (PRD) Revision 4.0, 2014.
[25] F. Chen et al. 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, April 2012. doi:10.1109/IRPS.2012.6241865.