Methodology of Generating Dual-Cell-Aware Tests

Yu-Hao Huang*, Ching-Ho Lu*, Tse-Wei Wu*, Yu-Teng Nien*,
Ying-Yen Chen†, Max Wu†, Jih-Nung Lee† and Mango C.-T. Chao*
*Dept. of Electronics Engineering & Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan,
†Realtek Semiconductor Corporation, Taiwan

***Abstract*—This paper introduces a novel fault model, called the dual-cell-aware (DCA) fault model, which targets the short defects locating between two adjacent standard cells placed in the layout. A layout-based methodology is also presented to automatically extract valid DCA faults from targeted designs and cell libraries. The identified DCA faults are outputted in a format that can be applied to a commercial ATPG tool for test generation. The result of ATPG and fault simulation based on industrial designs have demonstrated that the DCA faults cannot be fully covered by the tests of conventional fault models including stuck-at, transition, bridge and cell-aware faults and hence require their own designated tests to detect.**

I. Introduction

Fault modeling is the fundament of structural test. The conventional fault models, such as stuck-at (*SA*), transition delay (*TD*) and bridging (*BR*) faults, are well supported by current commercial ATPG tools and their resulting tests have been proved effective for detecting majority of manufacturing defects, especially those locating at or between the I/O ports of standard cells where SA, TD and BR faults are defined at. In order to cover those defects locating inside standard cells, a significant amount of research effort has been put in the past to develop new fault models or test metric for the applications requiring extremely low DPPM. Such a demand of close-to-zero DPPM has been significantly increased recently with the emergence of automotive devices.

The N-detect metric [1] [2] and the gate exhaustive (*GE*) metric [3] [4] are two classic test metrics for increasing the defect coverage. The N-detect metric attempts to detect a conventional SA or TD faults with N different patterns, meaning that N chances are created to randomly activate an un-modeled fault relating to the original simple fault and then propagating the un-modeled fault through the propagation paths of the original simple fault. The GE metric attempts to apply every possible input combination to each gate and observe the corresponding gate output. Achieving full coverage of the GE test metric for a standard cell can guarantee that all possible cell-internal defects are detected. Results based on silicon chips have demonstrated that using either N-detect or GE metric can help to detect various types of defects [5] [6] [7]. However, both N-detect and GE metrics are indirect models for detecting real defects and hence may generate more patterns than necessary.

Instead of using indirect models, another research line focuses on modeling the open defects inside a standard cell. Majority of cell-internal open defects can be successfully represented by transistor stuck-open (*TSO*) fault [8], which is similar to TD faults requires a 2-time-frame test pattern to detect. Applying TD faults for ATPG can cover part of TSO faults [9]. Applying TD faults with some additional constraints for ATPG can further improve the capability of a TD test for detecting TSO faults on primitive cells [10] and complex cells [11]. Besides TSO faults, cross wire open (*CWO*) faults [12] can be used to model the open defects locating in between two groups of parallel-connected pMOS or nMOS transistors on complex cells, which are not covered by the above TSO fault tests.

In order to cover more types of cell-internal defects, the cell-aware (*CA*) methodology [13] [14] was proposed to explicitly SPICE-simulate the faulty behavior of a cell with each potential defect injected to the RC netlist extracted from the cell's layout and then generate corresponding test patterns according to the fault-activation conditions obtained from the simulation. The types of injected defects for simulation include transistor stuck-on, transistor stuck-open, resistive open and short. The resulting CA faults can be a 1-time-frame (*1tf*) or 2-time-frame (*2tf*) fault. Results based on several production lines have demonstrated that the CA tests can effectively catch some defective parts not covered by other conventional tests [15] [16] [17]. This CA methodology is currently supported by a commercial ATPG [18] with a flexible format, named user-defined fault model (*UDFM*), that can describe complicated fault-activation and fault-observation conditions.

To further extend the coverage of defect types for the CA methodology, we in this paper propose a framework that can model the *dual-cell-aware* (*DCA*) faults occurring between two adjacent cells in the layout. A DCA fault can be extracted by SPICE-simulating the faulty behavior of an injected short defect between internal signals or I/O ports of two adjacent cells, which can be a 1tf or 2tf fault. This DCA fault extraction is also layout-based, meaning that an injected short defect must be located between two nets close enough in the layout and the RC netlists used for simulation are also extracted from cells' layout. The extracted DCA faults will be represented in the UDFM format and their corresponding DCA tests can then be generated by the commercial ATPG [18]. The experimental results based on industrial designs demonstrates that a significant portion of the DCA faults cannot be detected by the conventional tests, such as the SA, TD, BR and CA tests, and hence requires its own specialized tests to cover.

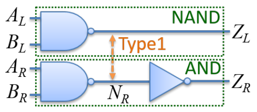
The remaining of this paper is organized as follows. In Section II, we first explain the concept of a dual-cell-aware fault and why it cannot be covered by the previous fault models. In Section III, we introduce our proposed framework for extracting DCA faults and how this framework can be integrated with the industrial design flow and commercial tools. In Section IV, we show the results of ATPG and fault simulations for applying different tests to 5 industrial block designs. The conclusion is given in Section V.

II. Dual-Cell-Aware Fault

DCA (dual-cell-aware) faults target the short defects between a pair of two adjacent cells placed in the layout. Such a pair of cells is called a *dual cell*, formed by its left cell L and right cell R. A short defect is injected between a node of the left cell and a node of the right cell, if these two nodes are close enough in the layout. In this section, we will introduce three types of DCA faults that cannot be modeled by any of the conventional SA, TD, BR and CA fault models.

A. Type-1: Port vs. Net before Inversion

A Type-1 DCA fault results from a short defect between a cell's I/O port and the other cell's internal net at the input of the output inverter. Fig. 1(a) illustrates an example of a Type-1 DCA fault, where the left cell is a NAND with two inputs A_L and B_L and one output Z_L and the right cell is an AND with two inputs A_R and B_R and one output Z_R. The AND on the right is formed by an NAND connected to an



(a) Location of short defect.

Z_L	N_R	Z_R
0	1/0	0/1
1	0/1	1/0

(b) 1tf fault.

Z_L	N_R	Z_R	Z_L	N_R	Z_R
R/0	0	1	0	R/0	F/1
F/1	1	0	1	F/1	R/0

(c) 2tf fault.

Fig. 1: A Type-1 1tf and 2tf DCA fault.

output inverter. In this example, the short defect is located between Z_L and the node N_R connecting to the input of the output inverter of the right cell.

We first discuss the case that the driving strength of Z_L is much stronger than that of N_R, which is a likely case since the signal strength of an I/O port is usually stronger than that of an internal net. As a result, the value at N_R is dominated by the value at Z_L, which can result in a 1tf fault with faulty response at Z_R. Fig. 1(b) lists the activation conditions for this 1tf fault, where Z_L must be different from N_R, meaning that Z_L must be equal to Z_R.

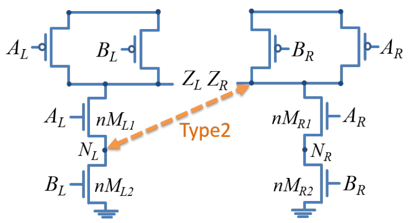
We then discuss the case that the driving strength of Z_L is similar to that of N_R. In this case, the values at both Z_L and N_R remain correct after DC analysis, but the signal delay of both Z_L and N_R may become longer when Z_L and N_R have opposing values, resulting in a 2tf fault. Fig. 1(c) lists the activation conditions for this 2tf fault, where a faulty transition can appear at either cell's output when the other cell's output value is set the same as the fault-free value of the final time frame at the faulty output transition. In other words, the condition for the final time frame is the same as the 1tf activation condition shown in Fig. 1(b) while the condition for the initial time frame is to create a proper transition at the targeted faulty output. Note that we consider only the 2tf patterns with a single-input transition in this paper.

Because activating above DCA faults requires the setting of input conditions for two cells at once, such faults cannot be modeled by any of SA, TD or CA model, which operates on one single cell. Also, the above DCA faults cannot be modeled by BR model either because the above DCA fault requires a pattern assigning the same value to both outputs of the two cells while a BR fault always requires a pattern assigning opposing values to the I/O ports of the two cells. This is the limitation of assuming that the fault can only occur at the I/O ports of cells, just as BR model does.

B. Type-2: Port vs. Internal Net

A Type-2 DCA fault results from a short defect between a cell's I/O port and the other cell's internal net inside the MOS network, which usually requires a more complicated activation condition than Type-1 DCA fault. In Fig. 2(a), we use a simple dual cell formed by two 2-input NANDs to illustrate the concept of a Type-2 DCA fault. In practice, the fault can occur on two more complicated cells. As Fig. 2(a) shows, the short defect of the exemplary Type-2 DCA fault locates between N_L, the node between the two serially connected nMOS transistors of the left cell, and Z_R, the output port of the right cell.

If the driving strength of the right cell is much stronger than that of the left cell, the value of N_L is dominated by Z_R, which can result in a 1tf fault with faulty response at Z_L. To activate this fault, Z_L and Z_R need to have opposite values and the nMOS nM_{L1} must be turned on to create connection between Z_L and Z_R. Fig. 2(b) first lists the activation conditions of this 1tf DCA fault. Next, Fig. 2(b) also lists the activation conditions of two 4-way BR faults with faulty response at Z_L, Z_R dominating Z_L at 0 (denoted as $Z_R \rightarrow Z_L(0)$) and Z_R dominating Z_L at 1 (denoted as $Z_R \rightarrow Z_L(1)$). As Fig. 2(b) shows, the two 4-way dominant BR faults cannot fully represent the DCA fault. Only one of the three conditions for $Z_R \rightarrow Z_L(0)$ can activate the DCA fault while $Z_R \rightarrow Z_L(1)$ can only describe one of the two conditions for activating the DCA fault.



(a) Location of short defect.

Fault	Z_R	A_L	B_L	Z_L
DCA	0	1	0	1/0
	1	1	1	0/1
BF $Z_R \rightarrow Z_L(0)$	0	0	0	1/0
	0	0	1	1/0
	0	1	0	1/0
BF $Z_R \rightarrow Z_L(1)$	1	1	1	0/1

(b) Activation conditions.

Fig. 2: A Type-2 1tf DCA fault and its activation condition.

If the left and right cells have similar driving strength, it will result in a 2tf DCA fault where the extra delay increase can occur at either cell. Fig. 3(a) and 3(b) lists the conditions of propagating the faulty transition through Z_L and Z_R, respectively. Following the same idea of generating the 2tf activation condition as in Section II-A, the condition of the final time frame of this 2tf DCA fault is the same as that of the 1tf DCA fault shown in Fig. 3 while the condition for the initial time frame is to create a proper transition at the targeted faulty output. Again, such complicated activation conditions cannot be fully represented by any previous conventional fault model.

A_R	B_R	Z_R	A_L	B_L	Z_L
1	1	0	1	F	R/0
0	X	1	R	1	F/1
X	0	1	R	1	F/1
0	X	1	1	R	F/1
X	0	1	1	R	F/1

(a) Faulty transition at Z_L

A_R	B_R	Z_R	A_L	B_L	Z_L
1	F	R/0	1	1	0
F	1	R/0	1	1	0
1	R	F/1	1	0	1
R	1	F/1	1	0	1

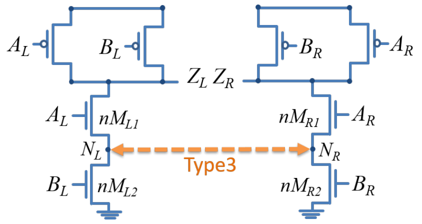
(b) Faulty transition at Z_R

Fig. 3: Activation condition for a Type-2 2tf DCA fault.

C. Type-3: Internal Net vs. Internal Net

A Type-3 DCA fault results from a short defect between two internal nets of the two cells. Fig. 4(a) shows an exemplary Type-3 DCA fault by using the same dual cell as in Fig. 2(a). In Fig. 4(a), the short defect is located between N_L and N_R, which are both the node in between the two serially connected nMOS transistors for either cell. If the driving strength of the right cell is much stronger than that of the left cell, the value of N_L is dominated by N_R, which can result in a 1tf fault with faulty response at Z_L. Fig. 4(b) lists the activation conditions of this 1tf DCA fault, where A_L needs to be 1 to turn on the nMOS nM_{L1} for connecting N_R with the targeted faulty output Z_L and the aggressor N_R needs to be the opposite value to the fault-free value at Z_L.

We also compare this Type-3 1tf DCA fault to the same two BR faults with faulty response at Z_L as shown in Fig. 2(b). For the case when $Z_L = 1/0$, only the last of the three activation conditions of $Z_R \rightarrow Z_L(0)$ can satisfy the activation condition of the Type-3 1tf DCA fault while the other two conditions cannot. For the case when $Z_L = 0/1$, the activation condition for the Type-3 1tf DCA fault is more strict than that of $Z_R \rightarrow Z_L(1)$, meaning that activating



(a) Location of short defect.

when N_R=0					
A_R	B_R	Z_R	A_L	B_L	Z_L
X	1	X	1	0	1/0

when N_R=1					
A_R	B_R	Z_R	A_L	B_L	Z_L
1	0	1	1	1	0/1

(b) Activation conditions.

Fig. 4: A Type-3 1tf DCA fault and its activation condition.

$Z_R \rightarrow Z_L(1)$ may not necessarily activate this Type-3 1tf DCA fault. The main difference here is that the Type-3 1tf DCA fault needs to consider the value of an internal net N_R while $Z_R \rightarrow Z_L(0)$ and $Z_R \rightarrow Z_L(1)$ only consider the values of I/O ports.

Due to the page limit, we omit the derivation of the activation conditions for the corresponding Type-3 2tf DCA fault, which can be obtained with the same method as used in Section II-A and Section II-B.

III. DCA Framework

Unlike CA methodology [13] [14] extracting CA fault models based on each individual standard cell, DCA fault models are extracted based on each pair of cells placed next to each other in the layout. Fig. 5 illustrates the overview of the proposed framework for extracting DCA fault models. First, the proposed framework identifies all possible pairs of adjacent cells that can form a valid dual cell in the targeted designs (Section III-A). Second, the proposed framework sets up the SPICE-simulation environment for each valid dual cell by merging the cell-library information (Section III-B) and creating RC netlist from the dual cell's layout (Section III-C). Next, proper short defects are injected into the RC netlist (Section III-D) and the faulty behavior of the short defect, including 1tf and 2tf faults, are simulated (Section III-E). Last, the DCA fault model of each valid dual cell is generated (Section III-F) and outputted in UDFM format (Section III-G). A SQL database is used in this framework to store and track the information of DCA fault models. All inputs and outputs of this framework follow the standard industrial formats in order to deal with industrial designs and cell libraries. The details of each step of the proposed framework are introduced in the following subsections.

A. Extraction of Valid Dual Cells

The number of all possible dual cells can be easily more than 10 millions since one design can be built based on multiple cell libraries with different V_t and channel length. To SPICE-simulate all possible dual cells with all possible defects is extremely computationally expensive. In order to control the simulation runtime, our framework only extracts the DCA fault models for the dual cells that are physically used in the targeted design. Note that the extracted DCA models for one design will be stored in the SQL database and can be reused when the next design contains the same dual cells. In other words, the number of dual cells to be simulated for a new design can be iteratively reduced as the DCA database keeps on growing with more designs applied.

In our framework, valid dual cells are extracted by examining the cell placement of the target design represented in the *.def* format. First, all pairs of cells placed next to each other in the design are identified. Next, we will filter out the pairs containing one of the following three invalid cells: (1) a spare cell, whose I/O ports cannot be controlled or observed, (2) a filler cell, which can be viewed as white space providing no functionality, and (3) a tie-cell, which can only result in a stuck-at fault on the other cell if a short occurs in between. Last, our framework will search the SQL database and eliminate the dual cells already included in the database. Remaining are the valid dual cells that will be used for generating DCA fault models for the targeted design. Note that our framework currently does not handle sequential cells, such as flip-flops and latches. All the dual cells are formed by two combinational cells.

B. Dual Cell Info Generator

In our framework, a dual cell can be virtually considered as a new standard cell formed by two existing standard cells and contains many inputs and multiple outputs, like a full adder. The objective of this step is to combine the information of the two cells into one based on the *.lib* files of the given cell libraries. The information of a dual cell includes the I/O direction and name of each I/O port, the

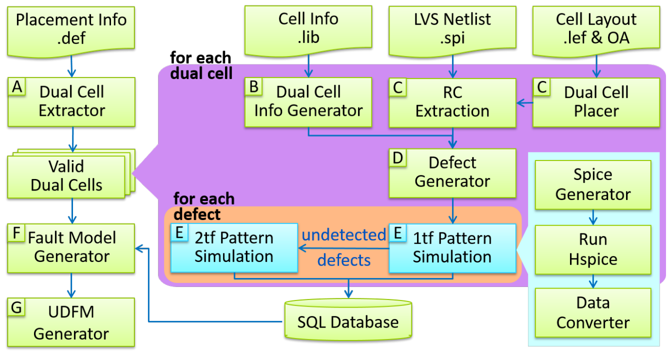


Fig. 5: Overview of proposed framework for extracting DCA faults.

function of each output and the driving strength. With the function of each output port, all 1tf and 2tf patterns along with the expected output responses can be generated for later defect simulation of a dual cell. Note that we consider the 2tf patterns with only single input transition and at least one output transition. Such a 2tf pattern has stronger capability of detecting transistor stuck-open faults for complex cells as recommended in [11]. Also, for a 2tf patterns with multiple input transitions, the transitions may arrive at the inputs at different timing in real application. It is highly time consuming to explicitly simulate the faulty behavior for each timing combination at inputs. It is even more computationally difficult at the ATPG stage to create the desired timing at inputs for activating the fault of a designated dual cell. That is why 2tf patterns with multiple input transitions are excluded for extracting 2tf DCA fault models in the proposed framework.

C. Dual Cell Placer and RC Extraction

Similar to CA methodology using layout-based analysis to generate fault models, our framework also performs the defect injection and defect simulation based on the RC netlist extracted from the physical layout, which is a more realistic way of characterizing the defect behavior. To generate the layout of each individual dual cell, we apply a commercial tool, Virtuoso [19], and use its SKILL script to automatically (1) place the two cells next to each other without any spacing in between, (2) label all pins according to the cell boundary and pin location defined in *.lef* file and (3) stream out to a GDS file. We set no spacing between the two cells because in real designs two adjacent active cells are either placed right next to each other with boundaries attached or separated with one or more filler cell. The RC netlist of a dual cell is extracted by a commercial, Calibre [20], with a LVS checking passed first based on the GDS file of the dual cell. The extracted RC netlist is outputted in DSPF (Detail Standard Parasitic Format).

D. Defect Generator

The objective of this step is to identify all possible short defects between the two single cells of a dual cell and then inject each short defect into the RC netlist for later defect simulation. We follow the same method used in the CA methodology [13] for identifying potential locations of injecting a short defect. In [13], a pair of nodes with a coupling capacitance in between is the location for injecting a short defect. A short defect can be injected by replacing the coupling capacitance with a defective resistance. The amount of the added defective resistance is a parameter set by users. In our later experiments, three different resistances, 10Ω, 1KΩ and 100KΩ, are used to represent different degree of shorting.

Note that at this step we only consider the cases that the two potential nodes for shorting are from different single cells. Otherwise, the short defect can be covered by a CA fault, instead of a DCA fault. We distinguish which single cell a node belongs to by comparing the naming of a node in the RC netlist. A prefix about its containing

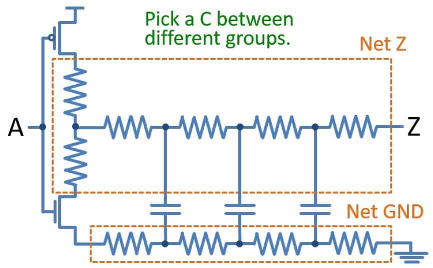


Fig. 6: Example of injecting a short defect in RC netlist.

single cell are preserved if we use sub-circuit to describe each single cell in the schematic file used for LVS checking of the dual cell.

In a RC netlist extracted from layout, one signal net can be broken into several resistance segments. If there are multiple coupling capacitances between two signal nets, only one of the capacitances will be chosen to inject short defect since the behavior of injecting a short defect to each of those capacitances is similar. Fig. 6 illustrates an example where only one of the three coupling capacitances between net_Z and net_GND will be used to inject short defect. This reduction of potential short-defect locations can help us to reduce the required simulation effort for generating DCA fault models.

E. Defect Simulation

Based on the defect-injected RC netlists extracted from the layout, the faulty behavior of each defect will be simulated by Hspice [21] and recorded in the SQL database for each dual cell. At this step, all possible 1tf patterns of a dual cell will be applied for each defect. The defects not detected by any 1tf pattern will be simulated with all possible 2tf patterns. By dropping the defects detected by 1tf patterns, the simulation time can be saved. After 1tf DC analysis of each defect, all the final steady-state output voltage at each output will be recorded for each 1tf pattern. After 2tf transient analysis of each defect, all the delay time (from 50% voltage at the input to 50% voltage at the output) at each output will be recorded for each 2tf pattern.

F. Fault Model Generator

The objective of this step is to generate the 1tf and 2tf DCA fault models for each valid dual cell. A DCA fault is defined as the set of all patterns that can detect a given defect. For 1tf DCA faults, a pattern can detect a defect if the pattern in the defect simulation can result in an output voltage deviating from its expected output voltage by more than a user-defined threshold, which is the same method as used in the CA methodology [13]. In our later experiments, this deviation threshold is set to 60%, the same as the default setting used in the commercial tool [22] for extracting CA fault models.

For 2tf DCA faults, a pattern can detect a defect if the pattern in the defect simulation can increase the delay time of an output transition by more than a user-defined threshold, which is different from the method used in the CA methodology [22]. This delay-increase threshold can be adjusted design by design according to the average cell slack or the distribution of the cell slacks reported by STA. One advantage here is that when changing to a different delay-increase threshold, we can simply re-examine the delay recorded in the SQL database and output new 2tf DCA faults without performing the time-consuming defect simulation again. In our later experiment, we set the delay-increase threshold to 500ps just for an example.

Fig. 7 shows the 1tf defect-simulation result of an exemplary dual cell formed by two single-input cells, where the input/output of the left cell L and the right cell R are denoted as A_L/Z_L and A_R/Z_R, respectively. Four exemplary defects $d1$, $d2$, $d3$ and $d4$ are used in Fig. 7 and the denotation D indicates that a defect can be detected by a pattern. In this case, three DCA faults will be generated, each for a detectable defect. The pattern set covering each fault is also

Pat.	Input	Output	Faulty Z_L			
	$A_L A_R$	$Z_L Z_R$	d1	d2	d3	d4
P_0	0 0	1 1	-	D	-	-
P_1	0 1	1 0	D	-	D	-
P_2	1 0	0 1	D	-	D	-
P_3	1 1	0 0	-	-	-	-

Fault1 (d1):
pattern set: (P_1 P_2)
Fault2 (d2):
pattern set: (P_0)
Fault3 (d3):
pattern set: (P_1 P_2)

Fig. 7: 1tf defect-simulation result of an exemplary dual cell.

listed in Fig. 7. The two faults with the same pattern set, Fault1(d1) and Fault3(d3), will be set as equivalent faults in the final outputted UDFM file to avoid redundant effort on ATPG and fault simulation.

G. DCA UDFM Generator

The extracted DCA faults will be outputted in UDFM supported by [18]. The DCA faults defined in UDFM are instance-based, meaning that if multiple instance of a dual cells exist in the target model, the DCA faults for each dual-cell instance needs to be specified individually. Fig. 8 shows an exemplary UDFM file which outputs the 1tf DCA faults of the same dual cell as used in Fig. 7. In this case, we assume there is only one dual-cell instance formed by the left-cell instance `L_inst` and right-cell instance `R_inst` in the module `BLOCK`. The UDFM in Fig. 8 first defines the module name. Then each DCA fault (declared by using `Fault`) is defined by specifying each of its patterns (by using `Test`). Each pattern is defined by specifying the faulty value at the output (by using `StaticFault`) and the condition at the inputs (by using `Condition`). An input or output used in this UDFM are associated with a designated instance in the module. A fault can be set equivalent to another fault by using `StaticEquivalent` followed by the name of the fault equivalent to, like the case that `Fault("d3")` is set equivalent to `Fault("d1")` in Fig. 8.

```
Module("BLOCK"){
  Fault("d1"){
    Test { //P1
      StaticFault{"L_inst/Z":0;} //Faulty value
      Conditions{"L_inst/A":0; "R_inst/A":1;} }
    Test { //P2
      StaticFault{"L_inst/Z":1;}
      Conditions{"L_inst/A":1; "R_inst/A":0;} }
  }
  Fault("d2"){
    Test { //P0
      StaticFault{"L_inst/Z":0;}
      Conditions{"L_inst/A":0; "R_inst/A":0;} }
  }
  Fault("d3"){ StaticEquivalent: "d1"}
}
```

Fig. 8: UDFM description of the 1tf DCA fault in Fig. 7.

IV. Experimental Result

A. Extraction of DCA Faults

The experiments in this paper are conducted based on five 28nm industrial block designs implemented by using the same four cell libraries with two different threshold voltages and two different channel lengths. Table I shows the result of DCA-fault extraction for each of these five designs, where the largest design contains 2M instances. Table I first lists the number of dual cells found in each design and the number of valid dual cells after filtering out all the invalid ones as shown in Section III-A. The ratio of the valid dual cells over the total found dual cells is listed next and ranges from 39% to 47% depending on designs. Table I then lists the average number of instances per dual cell. We hope that this number can

TABLE I: Result of DCA-fault extraction.

design	#inst.	#found dual cell	#valid dual cell	valid ratio	#inst. per dual cell	x-lib ratio	runtime	#dual cell simulated
D1	28K	6.0K	2.5K	42%	1.99	16%	6.0hr	2.5K
D2	45K	9.0K	4.3K	47%	2.13	28%	7.8hr	3.3K
D3	1.3M	43K	20K	47%	7.72	42%	48hr	20K
D4	2.0M	67K	31K	46%	5.35	61%	58hr	24K
D5	1.6M	37K	15K	39%	3.02	42%	22hr	9K
all					6.36		142hr	60K

be higher so that a simulated DCA fault can be reused on more instances. Table I also lists the ratio of dual cells formed by two cells from two different cell libraries, ranging from 16% to 61%. This result shows the importance of handling multiple cell libraries in a DCA framework.

Next, Table I lists the runtime spent on defect simulation as introduced in Section III-E, including both the 1tf and 2tf simulation. This runtime is the bottleneck of our DCA framework. In this experiment, the DCA-fault extraction is performed based on the order from D1 to D5 and our framework skips the simulation for the dual cells that have already been simulated previously. The reported runtime is obtained based on a computation system with 10 Hspice licenses. Table I further lists the number of dual cells actually being simulated for each design. As the result shows, total 60K dual cells are simulated for these five designs combined and the longest runtime for a single design is 58 hours. Note that the total number of possible dual cells that can be formed by the standard cells of these four cell libraries is 3.7M, which is 62.9X of our simulated dual cells. This is also the reason why our framework only simulates the valid dual cells existing in a design instead of exhaustively simulating all possible dual cells in advance.

B. ATPG Result

Two ATPG experiments are conducted in this subsection, one for 1tf faults and the other for 2tf faults as shown in Table II and Table III respectively. A commercial ATPG tool [18] is used for generating test patterns of each targeted fault model. In the 1tf-fault experiment, test patterns for (1) SA, (2) BR, (3) CA and (4) DCA fault model are generated serially in the given order. Before starting ATPG for a given fault model, a fault simulation is performed based on all the previously generated patterns and only the faults not detected by the previously generated patterns are used as the targeted faults for ATPG. The BR faults used for ATPG are created by adding all 4-way dominant BR faults to each critical pair of cell-external nets extracted by a in-house critical-layout analyzer. The CA faults in use are created by applying the commercial CA-extraction tool [22] with its default settings.

Table II first lists the number of faults, number of patterns, test coverage (denoted as *TC*) and the ATPG runtime for the generated test set of each 1tf fault model. The average TC for SA, BR, CA and DCA is 96.83%, 89.19%, 97.99% and 96.97%, respectively. Table II further lists the DCA fault coverage (denoted as *FC*) after adding the test patterns of each fault model into the fault simulation in the above given order. As the result shows, 82.30% of 1tf DCA faults in average can be detected by the SA tests with an average 5.4K pattern count, showing that majority of the testable 1tf DCA faults can be detected by SA tests. Then after adding BR tests into the pattern set, another 0.99% (total 83.29%) of 1tf DCA faults in average can be detected. This 0.99% increase in 1tf DCA coverage demonstrates that part of 1tf DCA faults can be still activated while activating BR faults, but this probability is still low when considering the average 3.5K pattern count for BR tests. Next, after adding 1tf CA tests, only another 0.10% (total 83.39%) of 1tf DCA faults in average can be detected. Even though the average 613 pattern count for 1tf CA tests is relatively smaller than that for BR tests (17.5% of the average BR pattern count), this coverage increase is still considered insignificant, showing that 1tf CA tests are not an effective alternate for detecting 1tf DCA faults. Last, after adding 1tf DCA tests, another 0.47% of 1tf DCA faults in average can be further detected by using average 361 designated patterns only, which is 10% of the average BR pattern count. This result demonstrates that some 1tf DCA fault can hardly be detected by the tests of conventional fault models and requires designated test patterns to detect.

In a similar format of Table II, Table III shows the result for the 2tf-fault experiment, where test patterns for (1) TD, (2) CA and (3) DCA fault model are generated serially in the given order. As the result shows, 46.72% of the 2tf DCA faults in average can be detected by TD tests with an average 6.1K pattern count. Then after adding 2tf CA tests, another 1.78% of DCA faults in average can be detected with average 2.8K patterns. Compared to the 1tf DCA FC of 1tf CA tests, the FC increase of 2tf CA tests is more significant (from 0.10% to 1.78%) while the 2tf CA pattern count is also increased significantly (from 212 to 2.8K). Last, after adding the 2tf DCA tests, another 4.01% of DCA faults in average can be further detected with average 356 patterns only. The average pattern count of 2tf DCA tests is only 12.7% of that of 2tf CA tests while its DCA FC increase is 2.3X of that of 2tf CA tests. This result demonstrates that, compared to 1tf DCA faults, 2tf DCA faults are even more difficult to be detected by the tests of conventional fault models and hence the advantage of including designated 2tf DCA tests is more significant.

In the experiment result reported by [17] shows a similar trend on CA tests. As [17] reported, the increase of 1tf CA coverage resulting from 1tf CA tests is 0.63% when compared to the original SA tests, which is less significant than the 3.83% increase of 2tf CA coverage resulting from 2tf CA tests when compared to the original TD tests. If compared to SA and TD tests, our 1tf and 2tf DCA tests can achieve an average 1.56% and 5.79% increase on 1tf and 2tf DCA FC, respectively. This result demonstrates that DCA faults are more difficult to be detected by conventional tests than CA faults.

C. Analysis of Activation Condition

We have built a program to examine whether the activation condition of a DCA fault can be equivalent to or dominating a fault of other fault models. The DCA faults used in this experiment are the ones extracted from the same industrial designs as used in Section IV-B. The SA, TD, BR and CA faults in use are the ones associated with the paired single cells in each dual cell. The BR faults in use are the ones locating between any two I/O ports in each dual cell. For each fault model, multiple equivalent faults are counted as one.

The upper and lower parts of Table IV show the results of the fault-model comparison for 1tf and 2tf DCA faults, respectively. Table IV first lists the number of 1tf faults used in this experiment for each fault model and their ratio to the number of the DCA faults. Next, Table IV lists the ratio of the DCA faults that are equivalent to or dominating a fault of a given fault model, respectively. As the 1tf result shows, no 1tf DCA fault is equivalent to or dominating a SA or 1tf CA fault. On the other hand, 12.6% and 73.9% of the DCA faults are equivalent to or dominating a BR fault, respectively. This result explains why the BR tests can result in a higher increase on 1tf DCA FC than that of the CA tests in Table II. However, the number of the BR faults here is 3346K, which is 11.6X of that of 1tf DCA faults. In other words, we need to generate tests for 11.6X of the 1tf DCA faults to cover 73.9% of them, which is not an effective way of detecting 1tf DCA faults. This result also explains why in Table II using average 3.5K BR patterns can create an average 0.99% increase on 1tf DCA FC while using only average 361 DCA patterns can create an average 0.47% increase on 1tf DCA FC.

As the 2tf result in Table IV shows, no 2tf DCA fault is equivalent to a TD or 2tf CA fault. Only 0.006% and 0.004% of the 2tf DCA faults are dominating the TD and 2tf CA faults, respectively. This is a negligible portion of 2tf DCA faults, especially when the number of TD and 2tf CA faults is 4.5X and 12.1X of that of 2tf DCA faults.

TABLE II: ATPG result for SA, BR, 1tf CA and 1tf DCA tests and their corresponding 1tf DCA fault coverage.

design	SA ATPG				BR ATPG				CA ATPG				DCA ATPG				DCA FC			
	#fault	#pat	TC	time	#fault	#pat	TC	time	#fault	#pat	TC	time	#fault	#pat	TC	time	SA	BR	CA	DCA
D1	208K	515	96.14%	3m	203K	649	85.52%	3m	747K	33	96.92%	6m	26K	62	93.02%	8m	76.18%	77.92%	77.97%	78.76%
D2	324K	745	97.23%	4m	287K	819	87.89%	6m	1.4M	92	98.27%	12m	49K	68	97.10%	11m	79.76%	82.16%	82.19%	82.72%
D3	9.6M	4.0K	96.39%	173m	5.0M	910	95.05%	24m	109M	505	97.72%	301m	863K	522	97.89%	303m	84.58%	84.83%	85.12%	85.37%
D4	14M	7.2K	99.13%	1013m	7.1M	2.0K	87.20%	65m	204M	255	99.58%	71m	913K	932	98.83%	691m	87.04%	87.21%	87.26%	87.53%
D5	11M	14K	95.26%	166m	7.5M	13K	90.30%	270m	160M	2.2K	97.45%	436m	179K	221	98.03%	44m	83.95%	84.33%	84.41%	84.94%
AVG	7.1M	5.4K	96.83%	272m	4.0M	3.5K	89.19%	73m	95M	613	97.99%	165m	406K	361	96.97%	212m	82.30%	83.29%	83.39%	83.86%

TABLE III: ATPG result for TD, 2tf CA and 2tf DCA tests and their corresponding 2tf DCA fault coverage.

design	TD ATPG				CA ATPG				DCA ATPG				DCA FC		
	#fault	#pat	TC	time	#fault	#pat	TC	time	#fault	#pat	TC	time	TD	CA	DCA
D1	72K	821	95.75%	2m	182K	329	94.57%	1m	5.5K	184	89.17%	4m	62.83%	65.12%	73.60%
D2	144K	1.4K	94.84%	4m	382K	535	95.09%	4m	14K	633	89.45%	10m	67.70%	70.06%	75.80%
D3	71K	1.6K	95.79%	20m	302K	408	93.46%	24m	3.5K	23	23.63%	153m	11.60%	11.89%	13.33%
D4	2.6M	2.0K	63.42%	403m	32M	2.0K	33.89%	1097m	204K	546	38.58%	21m	21.02%	22.98%	24.09%
D5	2.4M	25K	96.68%	106m	27M	11K	88.33%	445m	37K	395	85.16%	37m	70.44%	72.45%	75.73%
AVG	1.1M	6.1K	89.30%	107m	11.9M	2.8K	81.07%	314m	53K	356	65.20%	45m	46.72%	48.50%	52.51%

TABLE IV: DCA faults equivalent to or dominating other faults.

	model	#fault	ratio to #DCA fault	ratio of DCA faults	
				equivalent to	dominating
1tf	DCA	288K	1	100%	100%
	SA	723K	2.5	0%	0%
	BR	3346K	11.6	12.6%	73.9%
	CA	1534K	5.3	0%	0%
2tf	DCA	204K	1	100%	100%
	TD	923K	4.5	0%	0.006%
	CA	2469K	12.1	0%	0.004%

V. Conclusion

In this paper, we have presented a novel DCA (dual-cell-aware) fault model targeting the short defects between two adjacent standard cells in the layout. We have also developed a framework to automatically extract the DCA faults in a design and output the extracted DCA faults in the UDFM format, which can be imported by the commercial ATPG tool [18] for test generation. The experimental results based on the fault-model comparison and the ATPG results on industrial designs demonstrated that the DCA faults cannot be effectively covered by the tests of the conventional fault models supported by current commercial ATPG tools and hence require their own designated patterns to detect.

References

[1] S. C. Ma, P. Franco, and E. J. McCluskey, "An experimental chip to evaluate test techniques experiment results," in *Test Conference, 1995. Proceedings., International*, Oct 1995, pp. 663–672.

[2] I. Pomeranz and S. M. Reddy, "On n-detection test sets and variable n-detection test sets for transition faults," in *VLSI Test Symposium, 1999. Proceedings. 17th IEEE*, 1999, pp. 173–180.

[3] E. J. McCluskey, "Quality and single-stuck faults," in *Test Conference, 1993. Proceedings., International*, Oct 1993, pp. 597–.

[4] A. F. Lin, K. Y. Liao, K. Y. Chiang, and J. C. M. Li, "Target: Timing-aware gate exhaustive transition atpg for cell-internal defects," in *VLSI Design, Automation and Test (VLSI-DAT), 2015 International Symposium on*, April 2015, pp. 1–4.

[5] M. E. Amyeen, S. Venkataraman, A. Ojha, and S. Lee, "Evaluation of the quality of n-detect scan atpg patterns on a processor," in *Test Conference, 2004. Proceedings. ITC 2004. International*, Oct 2004, pp. 669–678.

[6] Y. T. Lin, O. Poku, R. D. Blanton, P. Nigh, P. Lloyd, and V. Iyengar, "Evaluating the effectiveness of physically-aware n-detect test using real silicon," in *2008 IEEE International Test Conference*, Oct 2008, pp. 1–9.

[7] K. Y. Cho, S. Mitra, and E. J. McCluskey, "Gate exhaustive testing," in *IEEE International Conference on Test, 2005.*, Nov 2005, pp. 7 pp.–777.

[8] R. L. Wadsack, "Fault modeling and logic simulation of cmos and mos integrated circuits," *The Bell System Technical Journal*, vol. 57, no. 5, pp. 1449–1474, May 1978.

[9] H. Cox and J. Rajski, "Stuck-open and transition fault testing in cmos complex gates," in *Test Conference, 1988. Proceedings. New Frontiers in Testing, International*, Sep 1988, pp. 688–694.

[10] N. Devtaprasanna, A. Gunda, P. Krishnamurthy, S. M. Reddy, and I. Pomeranz, "A unified method to detect transistor stuck-open faults and transition delay faults," in *Eleventh IEEE European Test Symposium (ETS'06)*, May 2006, pp. 185–192.

[11] X. Lin, W. T. Cheng, and J. Rajski, "On improving transition test set quality to detect cmos transistor stuck-open faults," in *2015 IEEE 24th Asian Test Symposium (ATS)*, Nov 2015, pp. 97–102.

[12] C. Han and A. D. Singh, "Testing cross wire opens within complex gates," in *2015 IEEE 33rd VLSI Test Symposium (VTS)*, April 2015, pp. 1–6.

[13] F. Hapke, R. Krenz-Baath, A. Glowatz, J. Schloeffel, H. Hashempour, S. Eichenberger, C. Hora, and D. Adolfsson, "Defect-oriented cell-aware atpg and fault simulation for industrial cell libraries and designs," in *2009 International Test Conference*, Nov 2009, pp. 1–10.

[14] F. Hapke, W. Redemund, J. Schloeffel, R. Krenz-Baath, A. Glowatz, M. Wittke, H. Hashempour, and S. Eichenberger, "Defect-oriented cell-internal testing," in *2010 IEEE International Test Conference*, Nov 2010, pp. 1–10.

[15] F. Hapke, J. Schloeffel, W. Redemund, A. Glowatz, J. Rajski, M. Reese, J. Rearick, and J. Rivers, "Cell-aware analysis for small-delay effects and production test results from different fault models," in *2011 IEEE International Test Conference*, Sept 2011, pp. 1–8.

[16] F. Hapke, M. Reese, J. Rivers, A. Over, V. Ravikumar, W. Redemund, A. Glowatz, J. Schloeffel, and J. Rajski, "Cell-aware production test results from a 32-nm notebook processor," in *2012 IEEE International Test Conference*, Nov 2012, pp. 1–9.

[17] F. Hapke, R. Arnold, M. Beck, M. Baby, S. Straehle, J. F. Goncalves, A. Panait, R. Behr, G. Maugard, A. Prashanthi, J. Schloeffel, W. Redemund, A. Glowatz, A. Fast, and J. Rajski, "Cell-aware experiences in a high-quality automotive test suite," in *2014 19th IEEE European Test Symposium (ETS)*, May 2014, pp. 1–6.

[18] *Tessent Scan and ATPG Users Manual*, v2014.1 ed., Mentor Graphics Corporation, March 2014, p.44-p.51.

[19] *Virtuoso Layout Suite L User Guide*, 6th ed., Cadence Design System, Inc., June 2015.

[20] *Calibre xRC Users Manual*, v2012.2 ed., Mentor Graphics Corporation.

[21] *HSPICE User Guide: Basic Simulation and Analysis*, Version j-2014.09 ed., Synopsys Inc., September 2014.

[22] *Tessent CellModelGen Tool Reference*, v2014.1 ed., Mentor Graphics Corporation, March 2014, p.215-p.220.