Tightening the Mesh Size of the Cell-Aware ATPG Net for Catching All Detectable Weakest Faults

Min-Chun Hu1,2,3 Zhan Gao1,2,4 Santosh Malagi2 Joe Swenton2
Jos Huiskens1 Kees Goossens4 Cheng-Wen Wu3 Erik Jan Marinissen1,4

1 IMEC
Kapeldreef 75
3001 Leuven
Belgium
min-chun.hu.ext@imec.be
zhan.gao.ext@imec.be
j.a.huisken@tue.nl

2 Cadence Design Systems
1701 North Street
Endicott, NY 13760
United States of America
minchun@cadence.com
zgaoo@cadence.com
malagi@cadence.com
swenton@cadence.com

3 National Tsing-Hua University
101, Section 2, Kuang-Fu Road
Hsinchu, 30013
Taiwan
minchunhu@gapp.nthu.edu.tw
cww@ece.nthu.edu.tw

4 TU Eindhoven
Kapeldreef 75
101, Section 2, Kuang-Fu Road
Hsinchu, 30013
Taiwan
min-chun.hu.ext@imec.be
zhan.gao.ext@imec.be

Conventional automatic test pattern generation (ATPG) approaches consider only faults on the interconnects between those cells [1]. Surely, their resulting test patterns also cover many cell-internal faults on a serendipitous basis, but as these intra-cell faults are not explicitly targeted by conventional ATPG, some remain undetected and cause test escapes. Eichenberger et al. [2] showed that a significant fraction of test escapes is caused by not-covered cell-internal defects. Preventing escapes is the main objective of manufacturing testing, as test escapes cause customers to become dissatisfied with the product quality.

It is widely accepted in the IC test community that manufacturing defects are accurately modeled as resistive shorts and opens [1]. In case of a low-ohmic short or a high-ohmic open, we classify the defect as hard. Variants of the same defect with higher (for shorts) or lower (for opens) resistance exhibit a less impactful fault behavior than hard defects and hence are called weak defects. In general, the weaker the defect, the more difficult it becomes to detect. In a digital cell-aware test, this typically implies that the number of cell-level patterns that detect an increasingly weaker defect variant diminishes until the defect is so weak that no patterns are left and the defect becomes undetectable. For a given defect, we refer to the weakest defect variant that is still detectable (at cell level) by means of a digital test as the weakest fault. In this paper, we propose a cell-aware test generation approach that explicitly targets the weakest fault at each potential cell-internal defect location.

We implemented this weakest-fault CAT (WF-CAT) approach in an experimental scripted tool flow, using various functions of Cadence’s Modus as building blocks. To assess the effectiveness of our approach, we formulate a new dedicated test metric: the weakest fault coverage \( wfc \). Compared to conventional CAT targeting hard defects only, experimental results show that our new approach enhances detection of weakest faults and significantly reduces \( wfc \) escapes \( \approx 1-wfc \), while maintaining its original (hard-defect) fault coverage \( fc \), of course at the expense of (acceptable) increases in the required number of test patterns and associated test generation time.

1 Introduction

Defects in integrated circuits (ICs) occur due to the large number of high-precision, defect-prone steps in their manufacturing process. To achieve sufficient quality for outgoing products, all ICs require testing for manufacturing defects to weed out defective parts prior to their shipment to customers. Unfortunately, these IC tests are not perfect either and occasionally let faulty ICs slip through the test: ‘test escapes’. Ballpark test escape rates are between 100 and 1 ppm (defective parts per million).

We observe an industry-wide push to improve the quality of IC structures are more sensitive to even very subtle defects; and (3) ICs are increasingly used in safety-critical applications (such as automotive, medical, avionics), and these markets demand the highest product quality with zero tolerance for test escapes.

Today’s digital IC designs are typically built up from interconnected standard cells taken from a pre-designed cell library.
2 Related Prior Work

CAT explicitly targets cell-internal open and short defects and thereby significantly reduces test-escape levels [3–5]. Cadence’s CAT approach is implemented as a two-step tool flow, coordinated by ATPG tool Modus, which invokes several other EDA tools along the way. In Step 1, the standard-cell library is characterized to determine per library cell which cell-level test patterns detect which cell-internal defects. First, we identify the possible defect locations in a library cell, assisted by parasitic extraction results from Quantus [6]. Subsequently, we assign a user-specified resistance value to the identified defects, and use analog simulator tool Spectre to determine for each library cell which defect is detected by which cell-level test pattern. The results are stored in a DDM per library cell [7]; this is a binary matrix where for library cell c, DDMc(d, p) = 1 if and only if defect d is detected by cell pattern p. The pattern set ps,c(d) of defect d is defined by ps,c(d) = {p ∈ Ps| DDMc(d, p) = 1}. Defects which are not detectable by any cell pattern (i.e., ps,c(d) = ∅) have apparently no effect on the function of the library cell, are classified as non-detectable and dropped from the DDMs; the remaining, detectable defects continue as faults. In Step 2, cell-aware ATPG tries to cover, for all cell instances in a design, all cell-internal faults by expanding the appropriate cell-level test patterns into chip-level test patterns (or into core-level test patterns, in case of modular SOC-based test development) [8].

Virtually all prior work on testing for resistive open and/or short defects does not relate to CAT [9–14], while prior work on CAT targets almost exclusively hard defects [3–5, 15, 16]. To the best of our knowledge, the only exception is a paper by Hapke et al. [17], in which CAT is used to try to detect both hard and weak short defects. The authors treat variants of the same defect with different resistive values as independent defects if their corresponding cell-level test patterns sets are non-identical. For a short defect s, consider two variants s1 and s2 with s2 weaker than s1. In case ps,c(s1) = ps,c(s2), [17] treats these two defect variants as one defect, but if ps,c(s2) ⊂ ps,c(s1), this single defect is modeled in [17] as two independent defects. If short defects are characterized for n different resistance values, a single defect can end up in the fault model of [17] as anywhere from one up to n independent faults. We consider these unpredictable effects on fault counts, and consequently on fault coverages, as undesirable.

Preliminary experiments on 350 cells from Cadence’s GPDK045 45nm CMOS library [18] show that if we vary the resistance value of the short defects in these library cells from 0 to 50kΩ, for about 60% of the short defects the cell pattern set capable of detecting the defect is independent of the short’s resistance value. However, for the remaining 40% of the shorts, the cell pattern set size varies as a function of the resistance value. If we consider per defect, the hardest and the weakest still-detectable resistance values, the defect model proposed in [17] would inflate the total number of short defects (11, 394) for no good reason by a factor 0.6 + 2 × 0.4 = 1.4 to 15,907 ‘defects’; considering 13 resistance values in the range [0, 30kΩ] (see Section 4), this unjustified inflation would even grow to 1.7 × 11, 394 = 19, 363 ‘defects’.

Unlike [17], in which with respect to weak defects only shorts are discussed, in this paper we address both short and open defects. We consider a weaker defect at the same location as just another variant of the same defect, thereby keeping the total defect count of a library cell a property of that cell only, not dependent on the number of resistance values considered during defect characterization. To avoid the anomalous fault coverage definition from [17], we introduce a new weakest-fault coverage wfc, next to the regular (‘hard’) fault coverage (fc).

3 Library Characterization

This section describes the WF-CAT library characterization, shown as Step 1 in Figure 1. We assume that a defect d with resistance r = R which is detectable by cell pattern p, remains detectable by p for all harder variants of d (i.e., for shorts with r ≤ R and for opens with r > R). This assumption implies that for each defect-pattern (d, p) combination for which there is a resistance value for which d is detectable by p, there is a weakest still-detectable resistance, which we refer to as the critical resistance of that particular defect-pattern combination.

Step 1 of our tool flow determines CRMs (or approximations thereof) for a standard-cell library C, for further usage in Step 2. On the basis of the cell layout and transistor-level netlist with extracted parasitics, we first identify for each library cell c ∈ C its set of potential defects Dc (type and location), as described in [6]. Subsequently, we need for each defect-pattern combination (d, p) with d ∈ Dc and p ∈ Pc (where Pc denotes the exhaustive cell-level pattern set for cell c) to determine its critical resistance CRM(d, p). This can be implemented through an iterative loop of Modus’ defect characterization function DC over appropriate ranges of resistance values. For a user-specified pair of short/open defect resistances rshort/ropen, DC determines for all defects d ∈ Dc and all cell patterns p ∈ Pc whether or not p detects d, denoted by a binary variable DDM(d, p) [7]. We could simply repeat DC for all possible resistance values in the defect resistance range considered. As Figure 2 shows, the resulting binary DDMs can be condensed into a single CRM, where for each entry (d, p), the critical resistance corresponds to the weakest resistance value rshort or ropen for which any of the DDMs denoted detection at (d, p).
DC is a compute-intensive operation. Fortunately, note that DC needs to be performed only once for each library release, while its results can be reused for all chip designs based on that library. This implies that we can afford to spend quite some compute time on DC. However, even a single DC iteration is expensive with respect to compute time, due to the fact that it repeats a detailed analog simulation in three nested loops: for all cells, for all defects, and for all cell patterns [4]. Adding a fourth loop ‘for all resistance values’ forces us to consider only a limited number n of defect resistances to avoid the required compute time to become excessively large.

First we determine the defect resistance ranges to be considered. The hardest defect resistance of each range should be super hard, such that we maximize the pattern set for each defect. The weakest resistances of each range should be such that for each defect-pattern combination, its critical resistance is included in the range. Hard shorts are low-ohmic and thus the natural candidate for the super-hard short is 0Ω. For super-hard opens and both weakest shorts and opens, the resistance values can vary per technology node and standard-cell design.

Next, the user needs to determine the number n of defect resistance pairs to be considered, and what their values will be. In our experience, a value of n for which the cumulative DC compute time is still considered affordable is too small to accurately perform DC for all possible critical resistance values; even at a granularity of, say, 1kΩ. This implies that both defect resistance ranges will be split into n bins that will approximate the critical resistance values.

Modus’ DC function needs as inputs (1) the transistor-level netlists of a set library cells C, (2) the set of short and open defects Dc for all c ∈ C, and (3) the resistance values rshort and ropen for respectively short and open defects. Its output is a binary DDM: for all c ∈ C.

In WF-CAT, defect characterization consists of n iterations of Modus’ DC function. It approximates for each library cell c the critical resistances for each defect-pattern combination (d, p) out of the iteratively generated DDM[c,i] (for i ∈ [1..n]) in a critical resistance bin matrix CRBM. If we order the resistance value pairs for the subsequent DC iterations from hard to weak (i.e., resistances increasing for shorts and decreasing for opens), then CRBMc is defined as follows:

\[
CRBM_c(d, p) = MAX_{i \in [1..n]}\{i | DDM_c[i](d, p) = 1\}
\]

Note that CRBM store DC iteration numbers, instead of actual resistance values; this allows us to treat short and open defects in an identical way. Translation from iteration numbers to user-defined defect resistance values is done through a two-dimensional look-up table R[t, i], where defect type t ∈ {short, open} and DC iteration i ∈ [1..n].

The n resistance values that determine the critical resistance bin boundaries do not need to be equidistant. It is best if these n values are chosen such that the division of the defects over the critical resistance bins is balanced.

Algorithm 1 presents our MULTI-R defect characterization algorithm, which iteratively calls Modus’ DC function to determine CRBMs for all cells in a library. Exploiting the fact that for subsequent DC iterations, the defect resistance values are ordered from hard to weak, we can for a defect-pattern combination (d, p) which in iteration i is still detectable simply overwrite the critical resistance bin so far by assigning \( CRBM(d, p) = i \) (Line 09). We can save significant compute time by pruning the number of defect simulations needed. Exploiting our assumption that all defects harder than the critical resistance remain detectable by the same cell pattern, as soon as for a particular iteration i a defect d ∈ Dc is no longer detectable for any pattern p ∈ P, we can drop that defect from subsequent DC iterations (Line 10). Similarly, if in a DC iteration a cell c ∈ C no longer has any detectable defects left, we can exclude that cell from subsequent DC iterations (Line 13).

Algorithm 1 [MULTI-R DC]

\[
\begin{align*}
\text{Inputs} & : C, D_c \text{ for all } c \in C; n, R[\{\text{short, open}\}, \{1..n\}] ; \\
\text{Outputs} & : CRBM_c \text{ for all } c \in C; \\
01: & i := 1; \\
02: & \text{while } i \leq n \land C \neq 0 \text{ do } \{ \\
03: & \text{DDM}_c[i] := DC(C, D_c, R[\text{short}, i], R[\text{open}, i]); \\
04: & \text{for all } c \in \text{library } C \text{ do } \{ \\
05: & \text{generate exhaustive cell-pattern set } P_c \text{ with } |P_c| = \#out \times 2^{|cm|}; \\
06: & \text{for all } d \in \text{defect set } D_c \text{ do } \{ \\
07: & \text{detectable } = \text{FALSE}; \\
08: & \text{for all patterns } p \in \text{pattern set } P_c \text{ do } \{ \\
09: & \text{if DDM}_c(d, p) = 1 \text{ then } \{ \\
10: & \text{detectable } = \text{TRUE}; \text{CRBM}_c(d, p) := i; \\
11: & \text{\}}; \\
12: & \text{if detectable } = \text{FALSE } \text{ then } D_c := D_c \setminus \{d\}; \\
13: & \text{if } D_c = \emptyset \text{ then } C := C \setminus \{c\}; \\
14: & \text{\}}; \\
15: & i := i + 1; \\
16: & \text{\}}; \\
\end{align*}
\]

Where Modus’ function DC has the following role:

\[
\begin{align*}
\text{Inputs} & : C, D_c \text{ for all } c \in C; r_{\text{short}}, r_{\text{open}}; \\
\text{Outputs} & : DDM_c \text{ for all } c \in C; \\
21: & \text{for all cells } c \in \text{library } C \text{ do } \{ \\
22: & \text{generate exhaustive cell-pattern set } P_c \text{ with } |P_c| = \#out \times 2^{|cm|}; \\
23: & \text{for all } d \in \text{defect set } D_c \text{ do } \{ \\
24: & \text{for all patterns } p \in \text{pattern set } P_c \text{ do } \{ \\
25: & \text{if FaultSimulation}(c + d, p) \neq \text{FaultSimulation}(c, p) \text{ then } \{ \\
26: & DDM_c(d, p) = 1; \\
27: & \text{else } DDM_c(d, p) = 0; \\
28: & \}\}\}
\end{align*}
\]

From the CRBMs that are generated by the WF-CAT library characterization, we extract two DDMs that play a prominent role in our WF-CAT ATPG approach (see Section 5). They are the weakest-fault DDM (WF-DDM) and the hardest-fault DDM (HF-DDM). This is illustrated by means of a small example in Figure 3. Figure 3(a) shows a CRBM corresponding to the CRM in Figure 2(b), and the tables in Figures 3(b) and 3(c) show respectively the extracted WF-DDM and HF-DDM. The HF-DDM denotes detection ‘1’ for all defect-pattern combinations for which the CRBM has a critical resistance recorded. In fact, the HF-DDM corresponds to the DDM as generated in DC iteration 1 of our MULTI-R DC algorithm and is based on two single defect resistance values for all defects: one for shorts and one for opens. This is different for the WF-DDM. The WF-DDM denotes detects only for those cell patterns that detect the weakest still-detectable variant of a defect. As the weakest still-detectable variant is defined per defect, the actual resistance values can vary for each defect. Note that all faults have at least one cell pattern that detects the weakest fault, but there can also be
multiple cell patterns that achieve this, as is the case for defect \(d_3\) in the example in Figure 3(b).

![Critical resistance bin matrix.](image1)

![Weakest-fault DDM.](image2)

![Hardest-fault DDM.](image3)

**Figure 3:** Example CRBM and corresponding WF-DDM and HF-DDM.

## 4 Library Characterization Results

We performed library characterization for 350 combinational standard cells from Cadence’s GPDK045 45nm CMOS library [18]. We used Modus v19.1, assisted by Quantus v18.1 for parasitic extraction during defect location identification and assisted by Spectre v19.1 for analog simulations during defect characterization. For the experiments reported in this paper, we limited ourselves to simulation of one-cycle cell-level test patterns (although the proposed tool flow can also handle two-cycle patterns).

For short defects, we set the defect resistance range to \([0 \Omega, 50k \Omega]\); \(0 \Omega\) is a rather obvious choice for a super-hard short, and through simulation we found that no short defect \(\geq 50k \Omega\) was detectable in any of the library cells. For open defects, initially, we considered to follow [17] and use \(1G \Omega\) as super-hard resistance. However, we found some defects that required an open resistance \(> 1G \Omega\) to become detectable; hence we used \(1,000G \Omega = 1T \Omega\) as safe super-hard resistance for opens. Finally, we ended up using 13 resistance bins; their resistance values are shown in Figure 4.

![Library Characterization Results](image4)

**Figure 4:** The number of defects per detect characterization iteration.

In Section 2 of this paper, we defined pattern set \(wps_c(d)\) as the set of cell patterns that detect defect \(d\) in cell \(c\). On the basis of that definition, we can now formulate the definitions for two specific pattern sets: \(hps_c(d)\) contains all cell patterns that are able to detect the hardest variant of defect \(d\), while \(wps_c(d)\) contains all cell patterns that are able to detect the weakest still-detectable variant of defect \(d\).

\[
\begin{align*}
\text{hps}_c(d) &= \{ p \in P_c | \text{DDM}_c[1](d,p) = 1 \} \\
\text{wps}_c(d) &= \{ p \in P_c | \text{CRBM}_c[1](d,p) = \text{MAX}_{p \in p_c} (\text{CRBM}_c(d,p)) \}
\end{align*}
\]

**Figure 5** shows both \(|\text{wps}_c(d)|\) and \(|\text{hps}_c(d)|\) for all 22,456 detectable defects in our library of 350 cells. By definition, \(\text{wps}_c(d) \subseteq \text{hps}_c(d)\), and thus \(|\text{wps}_c(d)| \leq |\text{hps}_c(d)|\), \(66.9\%\) of all detectable defects are resistance-independent faults, as their detecting pattern set does not depend on the defect resistance. For these defects \(|\text{wps}_c(d)| = |\text{hps}_c(d)|\) and hence \(\text{wps}_c(d) = \text{hps}_c(d)\). For these defects, regular CAT and WF-CAT achieve the same test quality. The other \(33.1\%\) of all detectable defects are resistance-dependent faults, for which \(\text{wps}_c(d) \subset \text{hps}_c(d)\) and hence \(|\text{wps}_c(d)| < |\text{hps}_c(d)|\).

![Patterns](image5)

**Figure 5:** The numbers of cell patterns for the hard and weakest faults.

To detect the weakest variants of those defects, it is important to select the right cell patterns. For all resistance-independent faults in the 350 GPDK045 library cells, on average \(|\text{hps}_c(d)| = 8.9\) patterns, while \(|\text{wps}_c(d)| = 3.5\) patterns. An ATPG tool that is unaware of these differences has a \((8.9 - 3.5)/8.9 = 60.7\%\) probability to select a cell pattern for expansion that does not detect the weakest fault(s). The maximum pattern set size difference \(|\text{hps}_c(d)| - |\text{wps}_c(d)|\) is 56 patterns for a defect in cell AOI33X1.

For each cell, we determined the required minimal number of
cell patterns that can cover all super-hard faults respectively all weakest faults. This problem is equivalent to the well-known NP-hard set cover problem and addressed by heuristic algorithm MinCover [7]. On average, hard faults can be detected by more cell patterns than weakest faults. Consequently, as shown in Figure 6, for each library cell more cell patterns are required to cover all weakest faults than to cover all hard faults. Over all characterized 350 cells, the accumulated required minimal number of cell patterns to cover all hard faults is 1,665, while 384 additional patterns (≈+23%) are needed to cover all weakest faults. On average, we require about one more cell pattern per library cell for covering all the weakest faults than we do for covering all hard faults.

![Figure 6: The minimal required number of cell patterns for covering all hard or weakest faults, compared to the cell-exhaustive pattern set.](image)

### 5 Weakest-Fault Cell-Aware ATPG

Cell-aware ATPG uses as inputs a cell-level netlist of the circuit-under-test and DDMs of all library cells instantiated in the netlist. For each cell instance, all detectable defects in the corresponding DDM are faults and serve as targets of the ATPG process. The ATPG tool tries to expand cell patterns from cell to chip level to cover as many as possible cell-internal faults. The circuit design surrounding a target cell might prevent the ATPG tool to successfully expand a cell pattern. Internal (unpublished) experiments done by us have shown that the chance to successfully expand an arbitrary cell pattern is ≈66%; and this is confirmed by what is reported as ‘gate-exhaustive fault coverage’ in [19].

At the end of Step 1, library characterization, we have generated two sets of DDMs for all library cells: WF-DDMs and HF-DDMs. The two DDMs have the same matrix dimensions, i.e., their defect/fault columns and cell pattern rows are identical. WF-DDMs and HF-DDMs only differ with respect to resistance-dependent faults, as in their defect-detecting pattern sets according to the WF-DDM matrix are true subsets of their HF-DDM counterparts. Modus’ cell-aware ATPG can be executed on the basis of either WF-DDMs or HF-DDMs.

Feeding exclusively WF-DDMs to the ATPG engine makes that the tool attempts to detect cell-internal faults only by means of those cell patterns that are able to detect the weakest still-detectable defect variant. With only WF-DDMs, we simply do not inform Modus that other cell patterns might be able to detect harder variants of the same defect. Cell-aware ATPG on the basis of WF-DDMs also causes the fault coverage reported by the ATPG engine to be the weakest-fault coverage (wfc). The wfc will end up lower than the regular fault coverage fc based on HF-DDMs, as the WF-DDMs contain less alternative cell patterns that each offer a chance to detect the target fault by successfully expansion to chip level. However, the benefit of WF-CAT ATPG is that by explicitly targeting weakest-faults, wfc is expected to be significantly higher than when cell-aware ATPG is performed on the basis of HF-DDMs only.

As shown in Step 2 in Figure 6, our WF-CAT ATPG tool flow consists of three stages. First, we focus on detection of the weakest faults, as detection of a weak fault implicitly guarantees detection of all harder variants of the same defect, while to opposite is not necessarily true. This is done by executing Modus cell-aware ATPG on the basis of WF-DDMs. A first set of test patterns $T_P$ is generated, along with their wfc. Test pattern set $T_P$ certainly also already provides significant regular fault coverage $fc$, but as Stage 1 focussed exclusively on maximizing wfc, it is possible that fc is not yet maximal. Therefore, in Stage 2 we determine $fc$ for $T_P$ by performing fault simulation on the basis of HF-DDMs. The still undetected hardest faults are than targeted in Stage 3, in which top-off ATPG on the basis of HF-DDMs maximizes $fc$ and generates an additional set of test patterns $T_P$.

### 6 ATPG Experimental Results

We performed ATPG withCadence’s Modus v19.1 on eleven circuits [20–22] that were mapped onto Cadence’s GPPDK045 45nm CMOS library [18]. For each circuit, we compare the regular CAT results versus our new WF-CAT results. In our experiments, we run the regular cell-aware ATPG with the HF-DDMs based on super-hard short defects of 0Ω and super-hard open defects of 1Ω. Results are presented in Table 1 in which circuits are sorted by increasing cell instance count. The table has six main columns: (1) circuit name, (2) design data, (3) weakest faults and the hard faults, each in three identical sub-columns, (4) still undetected hardest faults are than targeted in Stage 3, in which top-off ATPG on the basis of HF-DDMs maximizes $fc$ and generates an additional set of test patterns $T_P$.

<table>
<thead>
<tr>
<th>#inputs</th>
<th>#Patterns</th>
<th>Cell-Exhaustive Test</th>
<th>MinCover on WDDM</th>
<th>MinCover on HDDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>128</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

![Figure 6: The minimal required number of cell patterns for covering all hard or weakest faults, compared to the cell-exhaustive pattern set.](image)

Columns ‘Weakest FCov. Escapes (≈ 1 – wfc)’ and ‘Hard FCov. Escapes (≈ 1 – fc)’ present fault coverage escape data for the weakest faults and the hard faults, each in three identical sub-columns. With the term fault coverage escapes we refer to the fraction of non-covered faults. The first and second sub-columns give fault coverage escapes for resp. CAT and WF- CAT, while the third sub-column lists the relative reduction of the CAT fault coverage escape percentage by the WF-CAT fault coverage escape percentage. These two columns address the main key performance indicator for ATPG: test quality. The larger the reported reduction, the more effective our WF-CAT methodology. For regular CAT, wfc is determined by fault simulation of the
regular CAT test patterns on the basis on WF-DDMs. The results show that on average, WF-CAT reduces the weakest-fault coverage escapes (1 - \(wfc\)) with 59.7%. Column ‘Hard FCov. Escapes (= 1 – fc)’ shows the comparison of the hard fault coverage escapes between regular CAT and WF-CAT. The results show that our work on CAT achieves a higher fault coverage also on hard faults.

The last two columns present two other key performance indicators for ATPG: test execution time and test generation time. In column ‘\(\Delta T\) chip patterns’, we give the increase in test patterns required by WF-CAT in comparison to regular CAT. Similarly, the last column shows the increase of the ATPG compute time (including ATE, fault simulation, and top-off ATEPG) for WF-CAT. Both columns show that on average, to achieve 59.7% reduction on the weakest-fault coverage escapes, we need 24.7% more test patterns (i.e., more test execution time) and 69.1% more ATEPG compute time (i.e., test generation time) per chip design.

7 Conclusion

CAT based on only one resistance value for short defects and another one for opens always compromises the resulting test quality. If the considered defect resistance value is hard (weak), for a significant fraction of the potential defects this leads to incorrect, oversized (undersized) cell-pattern sets for weaker (harder) variants of those defects, and hence to overestimation (underestimation) of the actual fault coverage.

This paper presents WF-CAT, which targets per defect location the weakest-still-detectable defect variant, as its detection implicitly guarantees detection of all harder variants of the same defect, as well as a ‘super-hard’ variant of the defect. We have built an experimental tool flow on top of Cadence’s Modus CAT functions, consisting of two steps. Step 1 characterizes library cells, by first identifying their potential intra-cell short and open defect locations. Next, analog simulation for a set of defect resistance values determines the critical (= weakest still-detectable) resistance per defect-pattern combination, which is stored in a CR(B)M. Step 2, cell-aware ATEPG, uses the CR(B)M to target both the weakest as well as ‘super-hard’ faults, thus covering both ends of the detectable-defect resistance range. Defect characterization is time consuming, and even more so if performed for multiple resistances values. If defects are simulated from hard to weak, we can reduce simulation time significantly by dropping defects once they become undetectable. Experimental results for a set of eleven benchmark circuits show that on average WF-CAT reduces \(wfc\) escapes with 59.7% at a cost of 24.7% more chip-level test patterns and 69.1% more ATEPG compute time.

Acknowledgements – We thank the following persons for their support. At Cadence Design Systems: Vivek Chickermane, Anton Klotz, and Wolfgang Meyer. At IMEC: Kristof Croes, Dragonir Milojovic, Alessio Spessotto, Michele Stacci, and Ilirronen Tatar. Min-Chun Hu was during her stay at IMEC financially supported by the National Tsing-Hua University and IMEC. Zhan Gao is financially supported by the China Scholarship Council (CSC) and Cadence Design Systems in Endicott, NY, USA.

References