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

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

[1] IMEC
Kapeldreef 75
3001 Leuven
Belgium
min-chun.hu.ext@imec.be
zhan.gao.ext@imec.be
erik.jan.marinissen@imec.be

[2] Cadence Design Systems
1701 North Street
Endicott, NY 13760
United States of America
minchun@cadence.com
zgao@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@ee.nthu.edu.tw

[4] TU Eindhoven
Den Dolech 2
5612 AZ Eindhoven
the Netherlands
z.gao@tue.nl
j.a.huisken@tue.nl
k.g.w.goossens@tue.nl
e.j.marinissen@tue.nl

Abstract – **Cell-aware test (CAT) explicitly targets faults caused by cell-internal short and open defects and has been shown to significantly reduce test escape rates. CAT library cell characterization is typically done for only two defect resistance values: one representing hard opens and another one representing hard shorts. In this paper, similar to fishermen tightening the mesh size of their nets to catch small fish, we perform library characterization as efficiently as possible for a set of resistances representing increasingly weaker defects, and then adjust our ATPG flow to explicitly target faults caused by the weakest still-detectable variant of each potential defect. We implemented this novel approach in an experimental ATPG tool flow script, using 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 =*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 dppm (defective parts per million). We observe an industry-wide push to improve the quality of IC tests, with three main drivers: (1) IC designs integrate an ever-growing number of components, and hence the number of potential defect locations increases; (2) IC manufacturing processes are capable of printing ever-smaller features, and these smaller 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. 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 commercially-available Modus ATPG tool (release 19.1) as building blocks (see Figure 1). The dark-blue items in this figure are the new elements in this tool flow. In Step 1, per library cell, we identify the weakest still-detectable resistances for all defect-pattern combinations; we store them in a *critical resistance matrix* (CRM). In Step 2, these CRMs are used by the cell-aware ATPG engine to explicitly target the weakest still-detectable faults and generate test patterns for them.

The remainder of this paper is organized as follows. Section 2 summarizes related prior work. Section 3 introduces the library characterization flow of our proposed approach, in which now we simulate two ranges instead of two single resistance values. Characterization results for a 45nm CMOS standard-cell library are presented in Section 4. Section 5 describes our WF-CAT ATPG flow of which the explicit objective is to improve the wea-

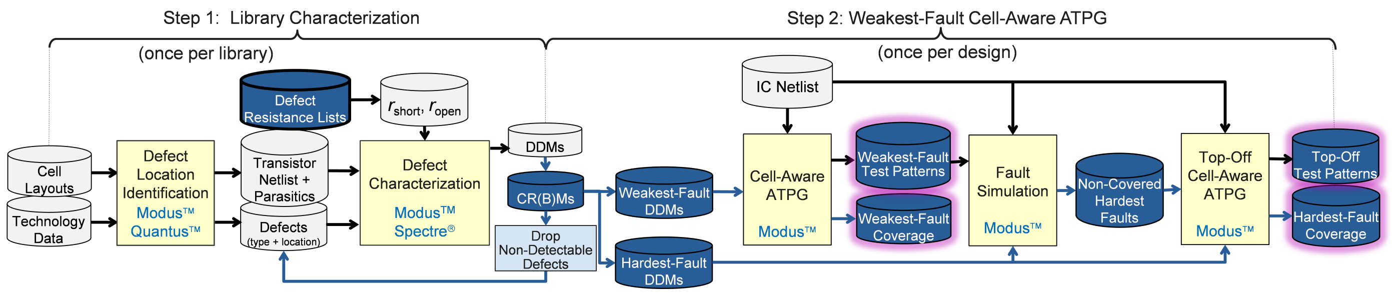

Figure 1: Weakest-fault cell-aware test (WF-CAT) tool flow (legend: 🟨 = CAT functions; ⬜ = CAT files; 🟦 = WS-CAT files; 🟦 = WS-CAT results).

kest fault coverage, while maintaining the original (hard) fault coverage of the conventional CAT approach. Section 6 presents weakest-fault ATPG results for eleven benchmark circuits and Section 7 concludes this paper.

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, $DDM_c(d,p) = 1$ if and only if defect d is detected by cell pattern p. The *pattern set* $ps_c(d)$ of defect d is defined by $ps_c(d) = \{p \in P_c | DDM_c(d,p) = 1\}$. Defects which are not detectable by any cell pattern (i.e., $ps_c(d) = \emptyset$) 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 s_1 and s_2 with s_2 weaker than s_1. In case $ps_c(s_1) = ps_c(s_2)$, [17] treats these two defect variants as one defect, but if $ps_c(s_2) \subset ps_c(s_1)$, 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 $0.6 + 2 \times 0.4 = 1.4$ to $15,907$ 'defects'; considering 13 resistance values in the range $[0, 30k)\Omega$ (see Section 4), this unjustified inflation would even grow to $1.7 \times 11,394 = 19,363$ '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 $c \in C$ its set of potential defects D_c (type and location), as described in [6]. Subsequently, we need for each defect-pattern combination (d,p) with $d \in D_c$ and $p \in P_c$ (where P_c 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 $r_{\text{short}}/r_{\text{open}}$, *DC* determines for all defects $d \in D_c$ and all cell patterns $p \in P_c$ 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 r_{short} or r_{open} for which any of the DDMs denoted detection at (d,p).

Pattern	**Short=0Ω**		**Open=1TΩ**	
`AB/Y`	d1	d2	d3	d4
p0 = `00/L`		1		
p1 = `01/L`	1	1		1
p2 = `10/L`	1		1	
p3 = `11/H`		1	1	1

(a) Multiple DDMs.

Pattern	**Short**		**Open**	
`AB/Y`	d1	d2	d3	d4
p0 = `00/L`		1k		
p1 = `01/L`	5k	5k		70M
p2 = `10/L`	1k		10.1M	
p3 = `11/H`		10k	10.1M	1G

(b) Critical resistance matrix (Ω).

Figure 2: Multiple DDMs for various open/short defect resistance value pairs can be condensed into a single CRM.

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 D_c for all $c \in C$, and (3) the resistance values r_{short} and (4) r_{open} for respectively short and open defects. Its output is a binary DDM_c for all $c \in C$.

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 $\mathit{DDM}_c[i]$ (for $i \in [1..n]$) 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 CRBM_c 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 resistances 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 $t \in \{short, open\}$ and DC iteration $i \in [1..n]$.

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 $\mathit{CRBM}(d, p) = i$ (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 $d \in D_c$ is no longer detectable for any pattern $p \in P_c$, we can drop that defect from subsequent DC iterations (Line 10). Similarly, if in a DC iteration a cell $c \in C$ no longer has any detectable defects left, we can exclude that cell from subsequent DC iterations (Line 13).

Algorithm 1 [MULTI-R DC]

```
Inputs   : $C$, $D_c$ for all $c \in C$, $n$, $R[\{short, open\}, \{1..n\}]$;
Outputs  : $CRBM_c$ for all $c \in C$;
01: $i := 1$;
02: while $i \leq n \wedge C \neq \emptyset$ do {
03:     $DDM_c[i] := DC(C, D_c, R[short, i], R[open, i])$;
04:     for all cells $c \in$ library $C$ do {
05:         generate exhaustive cell-pattern set $P_c$ with $|P_c| = \#out_c \times 2^{\#in_c}$;
06:         for all defects $d \in$ defect set $D_c$ do {
07:             detectable := FALSE;
08:             for all patterns $p \in$ pattern set $P_c$ do {
09:                 if $DDM_c(d,p) = 1$ then {
10:                     detectable := TRUE; $CRBM_c(d,p) := i$;
11:                 } };
12:             if detectable = FALSE then $D_c := D_c \setminus \{d\}$;
13:         } if $D_c = \emptyset$ then $C := C \setminus \{c\}$;
14:     }
15:     $i := i + 1$;
16: }
where Modus' function DC has the following role:
Inputs   : $C$, $D_c$ for all $C \in C$, $r_{\text{short}}$, $r_{\text{open}}$;
Outputs  : $DDM_c$ for all $c \in C$;
21: for all cells $c \in$ library $C$ do {
22:     generate exhaustive cell-pattern set $P_c$ with $|P_c| = \#out_c \times 2^{\#in_c}$;
23:     for all defects $d \in$ defect set $D_c$ do {
24:         for all patterns $p \in$ pattern set $P_c$ do {
25:             if $FaultSimulation(c + d, p) \neq FaultSimulation(c, p)$
26:                 then $DDM_c(d,p) = 1$;
27:                 else $DDM_c(d,p) = 0$;
28: } } }
```

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 detects: 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 $d3$ in the example in Figure 3(b).

Pattern AB/Y	Short d1	Short d2	Open d3	Open d4
p0 = `00/L`		4		
p1 = `01/L`	6	6		2
p2 = `10/L`	4		4	
p3 = `11/H`		10	4	1

(a) Critical resistance bin matrix.

Pattern AB/Y	Short d1	Short d2	Open d3	Open d4
p0 = `00/L`				
p1 = `01/L`	1			1
p2 = `10/L`			1	
p3 = `11/H`		1	1	

(b) Weakest-fault DDM.

Pattern AB/Y	Short d1	Short d2	Open d3	Open d4
p0 = `00/L`		1		
p1 = `01/L`	1	1		1
p2 = `10/L`	1		1	
p3 = `11/H`		1	1	1

(c) Hardest-fault DDM.

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

We identified 53,134 defect locations for the 350 library cells, i.e., on average 151.4 defects per cell. With the routine MULTI-R DC described as Algorithm 1 in Section 3 of this paper we performed defect characterization on these library cells with the abovementioned 13 defect resistance pairs. Figure 4 shows the number of defects considered per DC iteration. In the first iteration, all 53,134 defects were simulated for super-hard resistance values $r_{\text{short}} = 0\Omega$ and $r_{\text{open}} = 1T\Omega$. This required 18.1 days CPU time on high-end compute servers at Cadence in Endicott, NY, USA. 57.7% of the defects were classified as *non-detectable* by one-cycle cell patterns. The remaining 42.3% of the defects that were detectable in Iteration 1 continued in the second DC iteration with $r_{\text{short}} = 1\Omega$ and $r_{\text{open}} = 75M\Omega$. Figure 4 shows for all DC iterations th number of defects considered. With increasingly weaker defects in the subsequent DC iterations more defects become undetectable and once that happens, they are dropped from further DC iterations (as described in Algorithm 1). The overall compute time for MULTI-R DC was 65.6 days; quite expensive, but thanks to the defect and cell dropping mechanisms in Algorithm 1, the defect characterizations of 12 additional resistance values for all defects increased the compute time only 2.6×; this is a 72.2% reduction from what one would expect for 13 × 18.1 = 235.3 days. In our case, we were fortunate that the MULTI-R defect characterization could be performed on a multi-processor cluster, such that the elapsed wall-clock time was only 21 days.

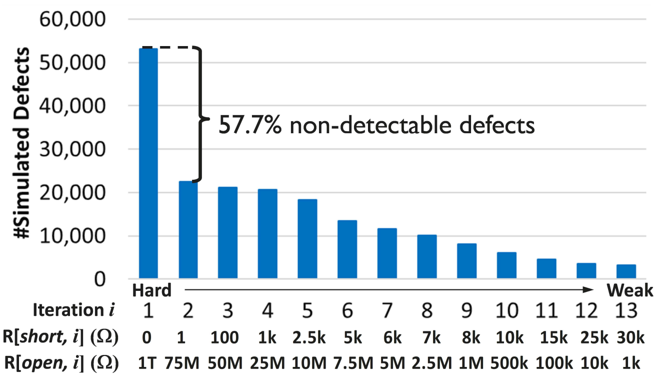

Figure 4: The number of defects per defect characterization iteration.

In Section 2 of this paper, we defined pattern set $ps_c(d)$ 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.

$$hps_c(d) = \{p \in P_c | DDM_c[1](d,p) = 1\}$$
$$wps_c(d) = \{p \in P_c | CRBM_c(d,p) = MAX_{p \in P_c}(CRBM_c(d,p))\}$$

Figure 5 shows both $|wps_c(d)|$ and $|hps_c(d)|$ for all 22,456 detectable defects in our library of 350 cells. By definition, $wps_c(d) \subseteq hps_c(d)$, and thus $|wps_c(d)| \leq |hps_c(d)|$. 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 $|wps_c(d)| = |hps_c(d)|$ and hence $wps_c(d) = hps_c(d)$. 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 $wps_c(d) \subset hps_c(d)$ and hence $|wps_c(d)| < |hps_c(d)|$.

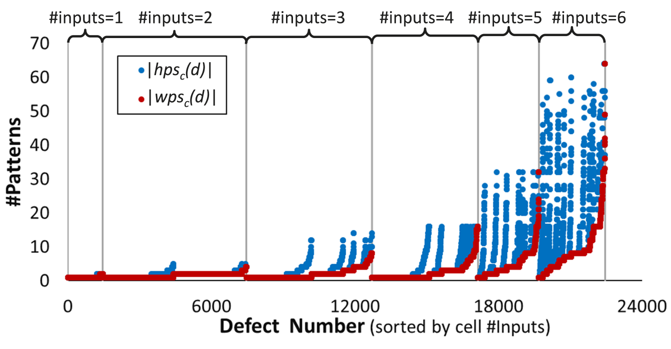

Figure 5: The numbers of cell patterns for the hard and weakest faults.

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 $|hps_c(d)| = 8.9$ patterns, while $|wps_c(d)| = 3.5$ 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 $|hps_c(d)| = |wps_c(d)|$ 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 $\mathcal{NP}$-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.

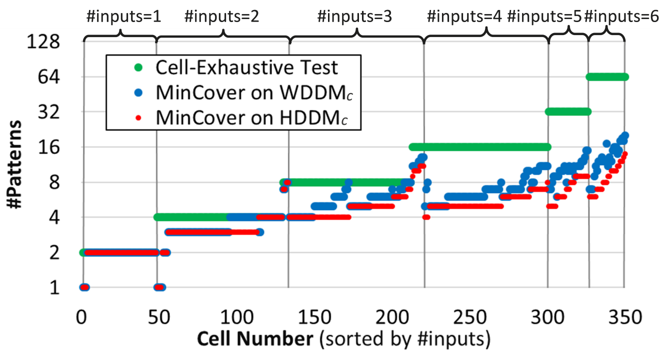

Figure 6: The minimal required number of cell patterns for covering all hard or weakest faults, compared to the cell-exhaustive pattern set.

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 , 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 TP_1 is generated, along with their *wfc*. Test pattern set TP_1 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 TP_1 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 TP_3.

6 ATPG Experimental Results

We performed ATPG with Cadence's Modus v19.1 on eleven circuits [20–22] that were mapped onto Cadence's GPDK045 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 1TΩ. 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 fault coverage escapes ($= 1 - wfc$), (4) hard fault coverage escapes ($= 1 - fc$), and the differences∆ between regular CAT and WF-CAT with respect to (5) chip test patterns and (6) ATPG compute time.

The column 'Design Data' consists of three sub-columns. In the first one, we list the number of cell instances $|I|$ in the circuit. In the second sub-column, we list the resistance-dependent faults F_{rd} as fraction of the total number of faults (see Equation (6.1)). In the third sub-column we list for the faults in F_{rd} the average difference between the numbers of hardest-fault and weakest-fault cell patterns, ΔP (see Equation (6.2)).

$$F_{\text{rd}} = \frac{\sum_{i \in I} |\{d \in D_{c(i)} | wps_{c(i)}(d) \subset hps_{c(i)}(d)\}|}{\sum_{i \in I} |\{d \in D_{c(i)} | hps_{c(i)}(d) \neq \emptyset\}|} \tag{6.1}$$

$$\Delta P = \frac{\sum_{i \in I} \sum_{d \in D_i} \left(|hps_{c(i)}(d)| - |wps_{c(i)}(d)|\right)}{\sum_{i \in I} |\{d \in D_i | wps_{c(i)}(d) \subset hps_{c(i)}(d)\}|} \tag{6.2}$$

In Equations (6.1) and (6.2), I is the set of cell instances, D_c is the set of defects for cell c, and $c(i)$ is a function that returns the cell type of cell instance i, with $i \in I$.

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

Circuit	Design Data			Weakest FCov. Escapes (=1-*wfc*)			Weakest FCov. Escapes (=1-*fc*)			Δ Chip	Δ ATPG
Name	$\lvert I \rvert$	F_{rd}	ΔP	CAT	WF-CAT	Δ(1-*wfc*)	CAT	WF-CAT	Δ(1-*fc*)	Test Patterns	Compute Time
b15 [20]	2,933	43.4%	4.54	2.00%	0.95%	-52.45%	0.49%	0.45%	-7.16%	+32.1%	+115.1%
b20 [20]	3,212	43.3%	3.52	0.56%	0.23%	-58.60%	0.17%	0.17%	0.00%	+15.2%	+61.7%
aes [21]	3,320	41.2%	4.70	0.41%	0.05%	-87.91%	0.04%	0.04%	-4.29%	+13.9%	+64.8%
b21 [20]	3,482	42.1%	3.33	0.62%	0.36%	-42.36%	0.29%	0.29%	0.00%	+14.7%	+60.9%
b22 [20]	5,036	43.0%	3.42	0.56%	0.27%	-52.11%	0.21%	0.21%	0.00%	+15.9%	+73.2%
M0 [22]	5,289	41.0%	5.74	0.81%	0.23%	-71.66%	0.16%	0.16%	-1.07%	+25.6%	+51.6%
b17 [20]	8,981	43.3%	4.58	1.79%	0.74%	-58.55%	0.44%	0.42%	-4.21%	+24.0%	+84.2%
fpu [21]	19,136	45.7%	4.66	0.29%	0.16%	-45.48%	0.14%	0.13%	-13.09%	+37.2%	+73.1%
b18 [20]	20,545	43.9%	4.55	1.52%	0.61%	-59.77%	0.36%	0.33%	-10.27%	+28.8%	+76.2%
M3 [22]	32,626	42.2%	5.60	1.03%	0.33%	-68.00%	0.22%	0.15%	-28.84%	+37.8%	+15.2%
b19 [20]	40,298	43.6%	4.53	1.50%	0.61%	-59.27%	0.37%	0.34%	-7.56%	+25.1%	+84.1%
Avg.	12,382	43.0%	4.47	1.01%	0.41%	-59.65%	0.26%	0.24%	-6.95%	+24.7%	+69.1%

Table 1: Comparison of CAT vs. WF-CAT ATPG results for 11 circuits w.r.t fault coverages *fc* and *wfc*, chip-level test pattern count, and ATPG compute time.

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 WF-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 'Δchip 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 ATPG, fault simulation, and top-off ATPG) 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 ATPG 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 (underachievement) 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 ATPG, 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 ATPG 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, Dragomir Milojevic, Alessio Spessot, Michele Stucchi, and Ibrahim 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

[1] M.L. Bushnell and V.D. Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Springer, Boston, MA, 2000. doi:10.1007/B117406.

[2] 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.

[3] F. Hapke et al. Defect-Oriented Cell-Aware ATPG and Fault Simulation for Industrial Cell Libraries and Designs. In *Proc. IEEE International Test Conference (ITC)*, pages 1–10, November 2009. doi:10.1109/TEST.2009.5355741.

[4] 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.

[5] 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)*, pages 1–10, October 2018. doi:10.1109/TEST.2018.8624906.

[6] Z. Gao et al. Defect-Location Identification for Cell-Aware Test. In *Proc. IEEE Latin-American Test Symposium (LATS)*, pages 1–6, March 2019. doi:10.1109/LATW.2019.8704561. (Best Paper Award).

[7] Z. Gao et al. Optimization of Cell-Aware ATPG Results by Manipulating Library Cells' Defect Detection Matrices. In *Proc. IEEE International Test Conference Asia (ITC-Asia)*, pages 91–96, September 2019. doi:10.1109/ITC-Asia.2019.00029.

[8] E.J. Marinissen. The Role of Test Protocols in Automated Test Generation for Embedded-Core-Based System ICs. *Journal of Electronic Testing: Theory and Applications*, 18(4/5):435–454, August 2002. doi:10.1023/A:1016545607915.

[9] T. Shinogi et al. Faulty Resistance Sectioning Technique for Resistive Bridging Fault ATPG Systems. In *Proc. IEEE Asian Test Symposium (ATS)*, pages 76–81, November 2001. doi:10.1109/ATS.2001.990263.

[10] M.T. Mohammadat et al. Resistive Open Faults Detectability Analysis and Implications for Testing Low Power Nanometric ICs. *IEEE Transactions on VLSI Systems*, 23(3):580–583, March 2015. doi:10.1109/TVLSI.2014.2312357.

[11] M. Arai, S. Inuyama, and K. Iwasaki. Layout-Aware 2-Step Window-Based Pattern Reordering for Fast Bridge/Open Test Generation. In *Proc. IEEE International Test Conference (ITC)*, pages 1–8, October 2017. doi:10.1109/TEST.2017.8242060.

[12] M. Andjelkovic et al. Impact of Resistive Open and Bridge Defects on the SET Robustness of Standard CMOS Combinational Logic. In *Proc. IEEE East-West Design Test Symposium (EWDTS)*, pages 1–6, September 2018. doi:10.1109/EWDTS.2018.8524748.

[13] G.C. Medeiros et al. Evaluating the Impact of Temperature on Dynamic Fault Behaviour of FinFET-Based SRAMs with Resistive Defects. *Journal of Electronic Testing: Theory and Applications*, March 2019. doi:10.1007/s10836-019-05784-1.

[14] A. Karel et al. Analytical Models for the Evaluation of Resistive Short Defect Detectability in Presence of Process Variations: Application to 28nm Bulk and FDSOI Technologies. *Journal of Electronic Testing: Theory and Applications*, 35(1):59–75, February 2019. doi:10.1007/s10836-019-05776-1.

[15] A.D. Singh. Cell Aware and Stuck-Open Tests. In *Proc. IEEE European Test Symposium (ETS)*, pages 1–6, May 2016. doi:10.1109/ETS.2016.7519316.

[16] S.P. Dixit, D.D. Vora, and K. Peng. Challenges in Cell-Aware Test. In *Proc. IEEE European Test Symposium (ETS)*, pages 1–6, May 2018. doi:10.1109/ETS.2018.8400700.

[17] 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.

[18] 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.

[19] 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.

[20] S. Davidson. ITC99 Benchmark Home Page. `https://www.cerc.utexas.edu/itc99-benchmarks/bench.html`, 1999.

[21] OpenCores.org. `https://www.opencores.org`, 1999.

[22] Arm Cortex-M Series Processors. `https://developer.arm.com/ip-products/processors/cortex-m`.