On the test of single via related defects in digital VLSI designs

On the test of single via related defects in digital VLSI designs

Nunzio Mirabella
AMS R&D
STMicroelectronics s.r.l.
Catania, Italy
nunzio.mirabella@st.com

Maurizio Ricci
AMS Data Storage & Custom PMICs
STMicroelectronics s.r.l.
Cornaredo (MI), Italy
maurizio.ricci@st.com

Michelangelo Grosso
AMS R&D
STMicroelectronics s.r.l.
Torino, Italy
michelangelo.grosso@st.com

***Abstract*—Vias are critical for digital circuit manufacturing, as they represent a common defect location, and a general DfM rule suggests replicating every instance for redundancy. When this is not achievable, a mandatory requirement is that the remaining single vias must be tested. We propose an automated method for generating tests and accurately evaluating test coverage of such defects, ready for use in any digital implementation flow and for integration within EDA tools, and also providing a useful quality metric. A prototype tool implementation and experimental results for an industrial case study are presented.**

Keywords—defect-oriented testing, single via, reliability, test, design for manufacturing

I. INTRODUCTION

Physical Design of digital integrated circuits (ICs) implies the placement of logic cells and the routing of signals and power nets. Routing is performed using a technology-dependent number of metal layers. Within a layer, planar horizontal and vertical connections are typically used (Manhattan routing). For a signal to get from one layer to a contiguous one, vertical connections known as *vias* are used (Fig. 1). Such vias are inherently hard to manufacture, i.e., defect probability in such structures is particularly high. As an example, Fig. 2 shows a microscope view of a defect in a 3µm technology, resulting from the outgassing from spin-on-glass (SOG) during via deposition and possibly causing an open or resistive connection.

Design for Manufacturing (DfM) rules strongly suggest duplicating every via instance in order to increase yield. However, in highly complex and integrated designs, via duplication percentage is often lower than the desired 100%, due to area limitations and routing congestion problems. This is especially relevant in technologies employed in mixed-signal and power electronics designs such as STMicroelectronics' Bipolar/CMOS/DMOS (BCD, [1]), where the coexistence of digital and high-voltage or high-power devices on the same substrate translates into limitations such as a reduced number of metal layers for routing with respect to purely digital, state-of-the-art devices.

One way of tolerating a double via percentage lower than 100% is to guarantee that every remaining single via is tested at the end of manufacturing. However, as of today, and to the authors' best knowledge, there is no publicly available tool for accurately evaluating the test coverage of via-related defects in an automated manner. This is a first requested step in order to develop a defect-oriented test generation flow.

This paper proposes an automated methodology for the evaluation of single via defect test coverage and for the development of a test set specifically targeting such defects. The proposed solution merges physical database and netlist information: each single via is automatically associated to one or more circuit nodes where a fault model instance (e.g., stuck-at or delay) is located, hence providing a fault list. Then, available Automatic Test Pattern Generation (ATPG) tools can be employed for generating the pattern set and/or for computing coverage of an existing test set. The 1-to-1 association between single via and fault location guarantees precise test coverage results addressing via-related defects.

The method is general in terms of adopted fault model and test application strategy (e.g., scan or functional). The identified faults represent a subset of the comprehensive circuit fault list. Even if a manufacturing test set aims at saturating the general test coverage, the information obtained on the single via defects is useful as a DfM metric, currently available within STMicroelectronics, and especially valuable when correlated with yield and quality data so as to monitor and enhance manufacturing processes.

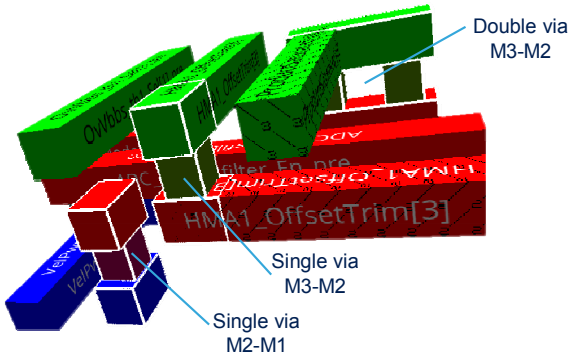

Fig. 1. Three-dimensional view of a portion of IC routing (obtained with Cadence Innovus). Three metal layers are visible, M1 (blue), M2 (red) and M3 (green), as well as two single vias and one double via.

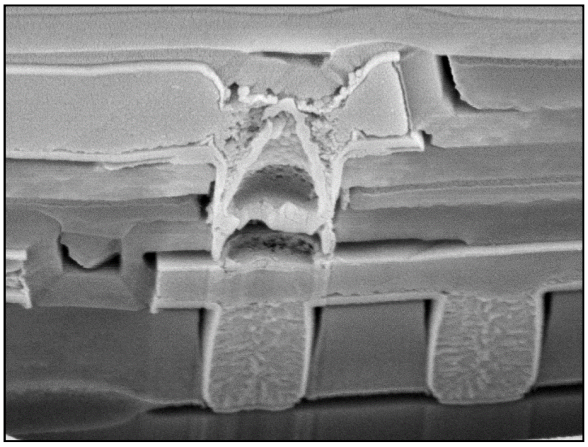

Fig. 2. Example of via defect (microscope view).

In addition, highlighting defect-prone areas and devices that may evolve into faults during the product mission life is useful for developing shorter test sequences that may be employed for fast screening (e.g., during Electrical Wafer Sorting, EWS) or in on-line test approaches in the field. When applying periodic test procedures, such as at each time a device is turned on, more precise coverage data can help in focusing the test strategy on the defects that are expected to appear first.

Finally, the methodology can also be used in iterative test generation or Place and Route (P&R) flows aimed at increasing manufacturability. After layout completion and test pattern generation, the testability of single vias can be evaluated, and then redundancy or test points may be selectively added on undetectable defect locations.

This paper is structured as follows: Section II provides an essential background on defect-oriented testing, via manufacturing and defectivity aspects, as well as the motivations of this work; Section III describes the proposed approach. Experimental results on a case study are reported in Section IV, and conclusions are drawn in Section V.

II. Background and Motivations

A. Defect-oriented testing

A defect is the unintended difference between the implemented hardware and its intended design, due to a physical imperfection in the processed wafer. Testing has the purpose of detecting manufacturing defects and process marginalities that cause circuit misbehavior (e.g., opens, shorts, threshold/parametric failures). Some defects are observable through visual inspection, but others are not visible and can only be detected by electrical tests. Defectivity has an impact on wafer yield, i.e., the average number of good chips produced per wafer, while testing also affects quality, representing the fraction of parts passing test (and thus shipped to customers) that are good. The goal of testing is to reduce the number of outgoing faulty parts.

The commonly used approach for test development in digital VLSI technology is based on modelling physical *defects* as logical *faults* on the circuit netlist or schematic. This approach enables the definition of test metrics (such as *coverage*) and the automation of test generation within a purely logic circuit representation. Fault models such as stuck-at and delay (path delay, transition, etc.) are commonly employed in today's VLSI design and manufacturing.

However, some studies show that traditional fault models are sometimes inadequate, and other methodologies have been developed and employed for specific technologies or to overcome occurring quality issues, using a *defect-oriented testing* approach [2][3][4]. Examples include dedicated test approaches for shorts, bridging faults and intra-cell defects [5][6][7].

In addition, defect-oriented testing can help in the determination of causes of failures, which is key for the optimization of specification, design, fabrication and test itself [8]. All is aimed to the improvement of defectivity and quality levels.

B. Vias, defectivity and fault modeling

A through-silicon, or through-chip, *via* is an electrical (metal) connection between different layers in a physical electronic circuit, which goes through the plane of one or more adjacent layers of a Si wafer and is electrically isolated from the substrate and from other via connections. Via manufacturing goes through various steps in the so-called damascene process: first, a dielectric layer is deposited, then trenches are etched according to the photoresist pattern and a barrier layer is deposited to stop metal diffusion into Si (typically Ta, TaN, TiN, or TiW). Metal (copper) is deposited and finally the surface is planarized using Chemical-Mechanical Planarization (CMP).

Vias in a VLSI design are among the most common sites where manufacturing defects occur and often cause reliability problems during operation. With interconnect width shrinking, increasing circuit density, and the introduction of copper metallization process, the probability of creating open vias or partially void vias has been increasing considerably [9]. As a matter of fact, metal voids are an inherent part of the process fabrication. Most voids do not cause any failures; however, current stress may cause voids to grow. Depending on the direction of current flow, a stress void can form on either side of the metal or via interface, possibly developing a resistive open fault. Defects can also occur as a result of local contamination, because of, for example, dust particles.

For these reasons, DfM guidelines recommend replacing single vias with double vias whenever possible, in order to exploit redundancy and to increase the margin against stress void failures, hence reducing defect probability, while lowering net resistance and improving signal delay. However, in highly dense circuits and especially in technologies with few available routing layers, the insertion of double vias can exacerbate routing congestion when performed during routing, or can lead to unsatisfying results if done at the end of P&R.

Any single via remaining in the circuit has a higher potential of being a defect location. For this reason, targeted tests and a specific test coverage evaluation for single via defects can provide valuable contributions, especially when correlated with manufacturing yield and product defect level data. In the literature, via defects are associated to open defects and usually modelled as stuck-at or, most often, delay faults [10][11]. The authors in [12] model single via defects with transition faults but do not provide details on the employed methodology for test generation and defect coverage computation. Few automatic test pattern generation tools specifically target open defects (e.g., [13]), while in [14] a method is proposed for associating the net position of the via defect in the netlist for failure analysis purposes.

C. Motivations

In order to automatically generate tests for defects on single vias, or to perform fault simulation to compute test coverage, it is necessary to associate defect positions to logic faults on a netlist. ATPG and fault simulation tools typically work only on the circuit netlist, so they are unaware of the location of any single vias. To the authors' best knowledge, there are no commercially available tools that allow merging physical and logical information in order to define a list of faults associated to single via defects in an automated way.

For each single via, it is not enough to know the net it belongs to, but also net branches have to be taken into consideration. Fig. 3.*a* presents the trivial case where the single via is on a net without any branches: in this case, the defect effect can be modeled as a fault on the net sink. A test for such a fault can detect the effects of the defect.

The example in Fig. 3.*b* shows the fact that it is not always necessary to test for faults on all the sinks of the net where the single via is present. Such a requirement would represent an overly strict requirement for the test, since the defect can be represented by either fault on A *or* B, and therefore it can be detected by any test covering the faults at just one of the two locations.

The cases in Fig. 3.*c* and Fig. 3.*d* are other examples of situations that can happen in real circuits. In some cases, layout-derived defect locations cannot be directly mapped to traditional fault locations. An example is a multi-fanout net, where some, but not all branches might be defective at the same time. This translates into the ATPG world as a multi-location fault, which is, to the best of our knowledge, not describable with commercial ATPG tools. To tackle the problem and automatically provide a list of faults for a commercial ATPG or fault simulator in such situations, the analysis of net structures is required, as described in the next Section.

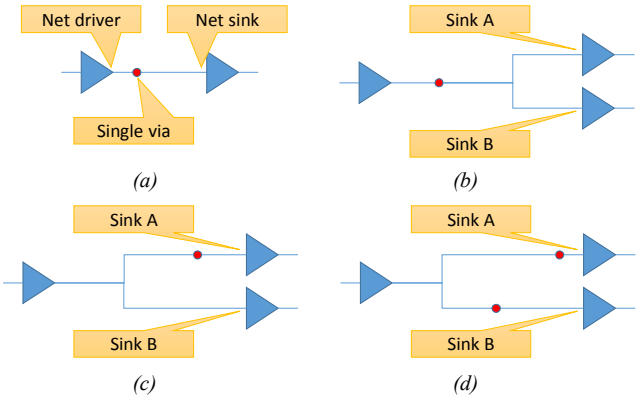

Fig. 3. Association between a single via and a fault position. In *a)*, the fault is placed on the only path sink. In *b)*, any test covering faults on sink A or sink B can detect the via defect. In *c)*, the fault on sink A is required, while in *d)* faults on sink A and sink B are related to different vias on the same net.

III. Proposed Approach

We propose an automated and practical methodology for generating specific tests (and for evaluating the test coverage) of single via related defects on digital circuits. This requires merging information about the layout and the netlist, to provide a fault list that can then be handled by traditional ATPG tools.

The flow is based on three main steps: first, the features of nets including a single via are extracted from the physical database. Then, the extracted data are analyzed in order to associate each single via (i.e., the defect location) to the node(s) in the netlist where the related fault is observable. Standard ATPGs or fault simulators can finally be used on the netlist to check the coverage of the faults on the identified nodes, and then the single via related defect coverage is evaluated.

A. Extraction of physical information

Fig. 4 shows the layout view of a sample net including a single via. Among the various metal components on different layers, the ones belonging to the target net are highlighted. In this case, the net has a driver, the output pin of a gate (consisting of the different metal rectangles encircled within the *out* ellipse), then bifurcates towards two other gate input pins. The connection towards the *in1* sink is on the same metal layer as the pins and uses two metal rectangles. The connection towards *in2*, instead, crosses a single via to get to a different metal layer, then passes through two metal rectangles and a double via to get to the sink pin.

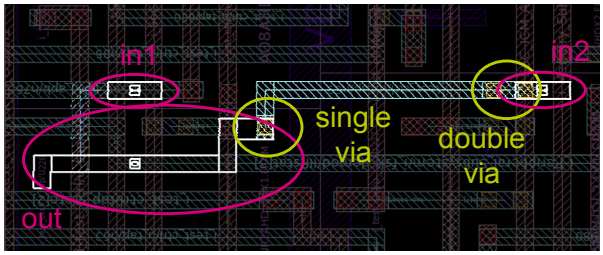

Fig. 4. Layout view of a net including a single via in the physical implementation tool (Synopsys IC Compiler).

This kind of information can be easily extracted for each net including a single via, from the physical database or from the physical design (PD) tool and expressed in a tool-independent format such as in the example in Fig. 5.

```
__NET i_test_ctrl/u_tap/n69
__PIN i_test_ctrl/u_tap/U249/B in metal1
{1527.520 775.920} {1527.800 775.920} {1527.800 776.820}
{1527.520 776.820} {1527.520 775.920}
__PIN i_test_ctrl/u_tap/U227/C in metal1
{1527.520 769.100} {1527.800 769.100} {1527.800 770.000}
{1527.520 770.000} {1527.520 769.100}
__PIN i_test_ctrl/u_tap/U226/Z out metal1
{1526.280 774.960} {1526.560 774.960} {1526.560 777.780}
{1526.280 777.780} {1526.280 774.960}
{1526.280 774.680} {1527.190 774.680} {1527.190 774.960}
{1526.280 774.960} {1526.280 774.680}
{1526.000 777.780} {1526.560 777.780} {1526.560 778.060}
{1526.000 778.060} {1526.000 777.780}
{1526.840 774.060} {1527.190 774.060} {1527.190 774.680}
{1526.840 774.680} {1526.840 774.060}
__WIRE VWIRE#32136289 metal1
{1527.520 776.820} {1527.800 777.440}
__WIRE HWIRE#33070350 metal1
{1526.280 777.160} {1527.800 777.440}
__WIRE HWIRE#33070351 metal2
{1526.900 774.060} {1527.800 774.340}
__WIRE VWIRE#32136290 metal2
{1527.520 769.980} {1527.800 774.340}
__VIA VIA#37365385 1 1 {metal1 metal2}
{1526.840 774.060} {1527.240 774.340}
__VIA VIA_ARRAY#33926197 2 1 {metal1 metal2}
{1527.520 769.660} {1527.800 770.580}
```

Fig. 5. Net elements description in tool-independent format.

The following data are reported:

- The hierarchical name of the *net* in the design;
- For circuit *ports* and gate *pins*, name of the instance, direction (in, out or inout), metal layer and shape(s);
- For *wires* (i.e., metal rectangular elements on a specific layer), name of the instance, metal layer and shape;
- For *vias* (which are represented as a bidimensional matrix of basic elements), name of the instance, number of rows and columns, top and bottom metal layers and shape.

Each of the element shapes can be expressed either as bounding boxes, i.e., the coordinates of the upper left and lower right corners in case of a rectangular item, or as polygons, consisting in the coordinates of the vertices of the composing rectangles (either with the last edge explicit, with the first and last point equal to each other, or implying the closing segment between the first and last point).

B. *Layout data analysis*

From the analysis of the previously extracted information, and in order to process data and obtain the association between the single via location and the fault (or faults) to be tested, a graph representation is used (*connection tree*). A node represents each one of the previously identified elements (ports and pins, wires and vias), while the presence of an edge indicates the physical contiguity, and hence the connection, between two elements (that can be computed from the shape and layer information). Fig. 6.*a* shows the graph deriving from the analysis of the previously introduced example.

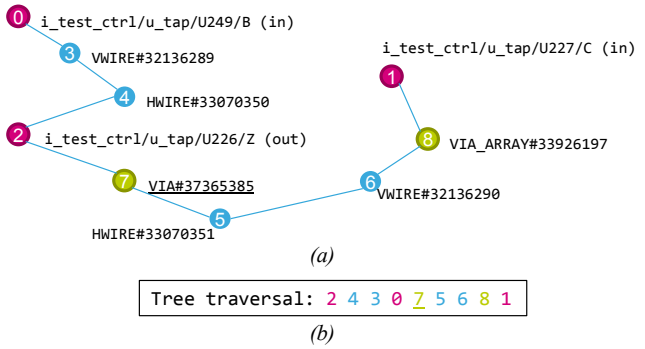

Fig. 6. In *a)* the representation of the net as a tree is shown; node 7 is the single via in the path. In *b)* the node sequence in the traversal is reported.

For each net, a depth-first traversal is used to associate each single via to the related fault location, starting from the net driver (tree root, node 2 in this case) towards all endpoints (Fig. 6.*b*). Along the traversal, each traversed single via and the endpoint reached are saved in a list. In this case, the single via at node 7 (`VIA#37365385`) is associated to the pin at node 1 (`i_test_ctrl/u_tap/U227/C`).

At the end of this step, the association between each single via and net sink(s) is stored in a *single via database*. From there, a fault list compatible with the selected ATPG (or fault simulator) tool is exported, deeming each fault as initially undetected. The fault model may be stuck-at, transition, small delay, or any other to model possible via defects in the addressed technology.

C. *ATPG/fault simulation and coverage evaluation*

The obtained fault list is used to perform ATPG or fault simulation of a previously generated test pattern set, using an available tool. The fault list is thus updated with the indication of coverage of each fault.

The last step requires comparing the data in the updated fault list with the single via database, to check which of the single via defects is tested by observing the coverage values on the related faults on endpoints. For each single via defect, at least one fault on its fanout needs to be covered to guarantee defect detection. From this, the final single via defect coverage is computed.

The flow chart in Fig. 7 resumes the complete flow.

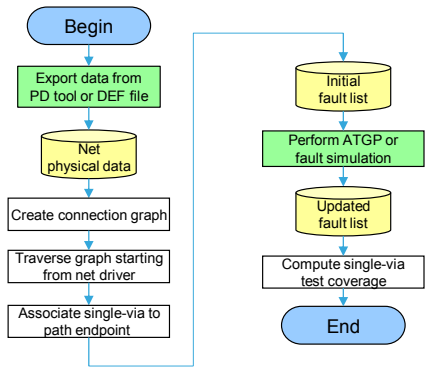

Fig. 7. Flow chart of the proposed methodology; green shapes imply the use of third-party tools.

D. *Discussion of special cases*

Among the different types of metal connections where a via can be present, there are some peculiar cases that have to be recognized and possibly handled in specific ways, on a case-by-case basis. Some of them may drive to real failures while others may be negligible and could safely be ignored and removed from coverage computation.

Among these, net segments connecting physical-only cells, such as those ending on antenna diodes, cannot be analyzed by ATPG, since they are not modelled in the circuit netlist. It may not even be possible to functionally detect defects on such nets due to controllability or observability limitations; however, in many cases such defects cannot cause a functional failure and thus belong to the redundant category, not impacting test coverage.

ICs typically integrate spare cells to provide a simple way to fix design errors in production with low-cost mask changes, affecting only the topmost metal layers. With their input pins tied to ground and their outputs unconnected, these cells cannot affect the functional behavior unless when a fix requires to modify the metal layers to connect them to the rest of the logic; if this occurs, the design changes and some spare cells become functional, but also test coverage must be re-evaluated and new test patterns have to be generated. Unused spare cell net connections will be reported as undetectable by ATPG, and, if considered, they will reduce the single via fault coverage. In addition, possible critical side defect effects can be detected, e.g., by quiescent current (IDDQ) testing.

Nets on the design boundary (e.g., connecting digital and analog parts in mixed-signal circuits, outside boundary scan structures) cannot be checked during scan test. Vias placed on these nets may lead to potentially dangerous failures which will not be systematically detected, so they should be considered during test coverage evaluation. Reporting these nets can be useful to develop dedicated functional tests, or non-default P&R rules can be defined to mandate redundancy with no exceptions in these cases.

Finally, nets directly connected to power and ground constitute more complex extraction and analysis cases, since they can have a very large number of composing elements and branches. Some single vias on these nets can be detected if used for tying cell pins to 0 or 1: these vias can effectively produce failures and should be considered during the analysis.

IV. Case Study

A. Prototype tool implementation

A prototype implementation of the developed methodology is currently available within STMicroelectronics. The toolset includes *tcl* scripts for extracting the design information from the physical implementation tool `export_geometry_<tool>.tcl` (Cadence Innovus, Synopsys IC Compiler and IC Compiler II) in a tool-independent format, a C++ analysis tool able to generate the fault lists in Synopsys TetraMAX format (`generate_faultlists.cpp`) and a C++ program for computing via defect coverage after fault simulation (`compute_coverage.cpp`).

The `export_geometry_<tool>.tcl` script creates a collection of all nets including at least one single via, and then iterates on those to extract the information for the composing elements of each net: *ports* and *pins*, *wires*, and *vias*, as described in III.A.

The C++ analysis tool `generate_faultlists.cpp` uses a base class *Element* and the derived classes *Pin*, *Port* and *Via* to collect and then process the information of each extracted net. The connection tree is built and stored in an adjacency list. For determining the physical contiguity among the geometric elements composing a net, the *distance* function from the open-source Boost.Geometry C++ library [15] was used, which enables working directly on polygons. Algorithm parallelization is straightforward, since each net can be analyzed at the same time as the others; therefore, the tool uses multithreading to speed-up computation. The tool produces a single via-fault association database and the fault lists for ATPG or fault simulation.

Finally, `compute_coverage.cpp` processes the post-ATPG list of faults (i.e., with each fault deemed as covered or not) and the previously generated database to tell for each via if the related defects are covered or not by the test patterns.

B. Experimental results

The flow has been experimented on different chips, some of them currently in development, and also on legacy ones to assess the performance of the test patterns used in production. This paragraph reports the results obtained on the digital subsystem of a mixed-signal IC implemented with Synopsys IC Compiler and manufactured in BCD8sP technology (160 nm feature size). This technology [16] combines the Bipolar process for precise analog functions, CMOS (Complementary Metal Oxide Semiconductor) for digital design and DMOS (Double Diffused Metal Oxide Semiconductor) for power and high-voltage elements.

The case study counts ~170k gate instances and has a placement area of about 5.1 mm^2. Full-scan testing is employed. The technology uses 4 metal layers for routing, and the circuit implementation has 1.6M vias. Even if the target double via percentage was set to 100%, in the back-end flow 95.07% redundancy was achieved, due to the position of macros and to the inherent technology limits. In total, there are 78,011 single vias, associated to 39,208 different nets. Out of those, 76,907 belong to signal nets and the other ones to power or ground tie connections which will be discussed later on. Fig. 8 shows the characteristics of the signal nets including single vias: it appears that there is a relevant quantity of "complex" nets that present several single vias and high fanout, which are relevant from an analysis point of view (the association between the logical fault position and the via is not straightforward).

76,813 single vias on signal nets were successfully associated by means of the previously described tools to circuit nodes, where to inject faults for ATPG. The other ones (94) are located on paths leading to unobservable points, such as antenna diode pins. Two fault lists were generated, one for stuck-at and one for transition faults, each counting 266,174 faults.

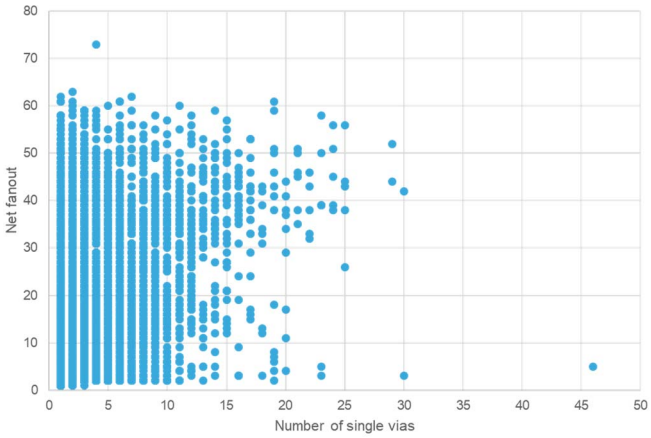

Fig. 8. Distribution of fanout and single via number for each analyzed signal net.

Signal net analysis (`generate_faultlists.cpp` compiled with GNU g++ 7.2 using the `-Ofast` option and resorting to 32 threads for computation) took 75 seconds on an Intel® Xeon® Gold 6154 CPU clocked at 3.00 GHz, with a maximum RAM occupation of about 3MB.

The two fault lists were used as a starting point for ATPG and Synopsys TetraMAX was used to generate scan test patterns. Setting the target coverage to 100%, the generated stuck-at test set was able to cover 98.23% of the fault list, while the transition test set coverage was 80.15%. These numbers correspond to 72,711 (94.7%) single via defect locations for stuck-at and 58,094 (75.6%) for transition faults.

Concerning the single vias on tie connections, 951 are on ground and 153 on power nets. Among the first ones, 921 are connected to ground supply pins or to spare cells inputs or belong to interconnect shielding, so are not observable and untestable from the ATPG point of view. Conversely, all single vias on power nets can be associated to logic cell input pins. In the end, 195 vias on tie nets can be associated to faultable nodes, and 153 defect locations can be detected by stuck-at tests, while the transition coverage is not relevant since the considered nets are intrinsically static. Due to their much larger number of composing metal elements (a few hundred thousand, whose mutual distance has to be computed to build the connection tree, compared to a few hundred elements of the most complex signal nets), the analysis of power and ground nets is slower and took a few hours with the current tool implementation.

It is worthwhile noting that using the patterns obtained with a generic ATPG flow on the whole digital subsystem reaching 99% stuck-at coverage, a fault simulation experiment confirmed the same results on the fault lists generated with the current approach without the ATPG specifically targeting the single via-related faults. This means that the ATPG maximizes coverage independently of the flow applied;

however, defect-prone structures may be prioritized in pattern generation in the case the pattern count is limited due to test cost or equipment constraints. The pattern set targeted to single via defects is about 20% smaller than the generic one.

Table I reports a summary of the obtained results.

TABLE I. SUMMARY OF RESULTS

Total vias	1.6M
Total single vias	78,011
Single vias on signal nets	76,907
Single vias on power/ground nets	1,104
Single vias associated to fault locations	76,813 power/ground 195
Single via defect locations tested (*stuck-at fault model*)	72,711 power/ground 153
Single via defect locations tested (*transition fault model*)	58,094

In summary, out of the initial 1.6M vias, 99.7% are either redundant or the defects they may cause are covered by stuck-at fault tests (98.7% considering transition faults). The remaining untested single vias are spread on all the placement area.

The obtained data is ready to be correlated with yield figures and can be the starting point for further test development (e.g., by means of functional techniques [17]) or circuit modifications to enhance redundancy of critical structures, possibly at the expense of additional area.

V. CONCLUSIONS

Manufacturing defects in digital circuits occur more likely in some specific structures, such as through-silicon vias, which undergo critical process steps. This paper has presented an automated methodology for generating tests aiming at detecting single via defects, and for precisely evaluating test coverage on them. A prototype tool is available within STMicroelectronics and it has been used on current and legacy products to provide a useful DfM metric. Experimental results were provided, showing the achievable level of detail and the immediacy of application in the design environment.

Future works aim at further exploiting the correlation of test coverage with yield and defectivity on industrial products, to improve design and manufacturing processes, and at exploring selective layout hardening flows based on the achievable results.

REFERENCES

[1] B. Murari, C. Contiero, R. Gariboldi, S. Sueri, A. Russo, "Smart power technologies evolution," IEEE Industry Applications Conference. 2000, pp. P10-P19 vol.1

[2] M.L Bushnell, V.D. Agrawal, "Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits," Springer, 2005

[3] M. Sachdev, "Defect Oriented Testing for CMOS Analog and Digital Circuits," Kluwer Academic Publishers, 1998

[4] F. Celeiro, L. Dias, J. Ferreira, M.B. Santos, J.P. Teixeira, "Defect-Oriented IC Test and Diagnosis Using VHDL Fault Simulation," IEEE International Test Conference, 1996, pp. 620-628

[5] J. Sudbrock, J. Raik, R. Ubar, W. Kuzmicz, W. Pleskacz, "Defect-oriented test- and layout-generation for standard-cell ASIC designs," IEEE Euromicro Conference on Digital System Design, 2005, pp. 79-82

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

[7] A. Bosio, L. Dilillo, P. Girard, A. Todri-Sanial, A. Virazel, S. Bernabovi, P. Bernardi, "An Intra-Cell Defect Grading Tool," IEEE Design and Diagnostics of Electronic Circuits and Systems, 2014, pp. 298-301

[8] Y. Sato, I. Yamazaki, H. Yamanaka, T. Ikeda, M. Takakura, "A persistent diagnostic technique for unstable defects," IEEE International Test Conference, 2002, pp. 242-249

[9] T.-Y. Lin, T.-H. Lin, H.-H Tung, R.-B. Lin, "Double-Via-Driven Standard Cell Library Design," IEEE Design, Automation & Test in Europe Conference & Exhibition, 2007, pp. 1-6

[10] K. Baker, G. Gronthoud, M. Lousberg, I. Schanstra, C. Hawkins, "Defect-Based Delay Testing of Resistive Vias-Contacts; A Critical Evaluation," IEEE International Test Conference, 1999, pp. 467-476

[11] R. Rodríguez Montañés, J. Pineda de Gyvez, P. Volf, "Resistance Characterization for Weak Open Defects," IEEE Design & Test of Computers, 19(5), September-October 2002, pp. 18-26

[12] D. Kim, M.E. Amyeen, S. Venkataraman, I. Pomeranz, S. Basumallick, B. Landau, "Testing for Systematic Defects Based on DFM Guidelines," IEEE International Test Conference, 2007, pp. 1-10

[13] S. Spinner, I. Polian, P. Engelke, B. Becker, M. Keim, W. Cheng, "Automatic Test Pattern Generation for Interconnect Open Defect," IEEE VLSI Test Symposium, 2008, pp. 181-186

[14] A. Ladhar, M. Masmoudi, "Extraction of Open-Via Defects from Industrial Designs," IEEE International Conference on Signals, Circuits and Systems, 2009, pp. 1-6

[15] Boost C++ libraries, https://www.boost.org/

[16] R. Roggero, G. Croce, P. Gattari, E. Castellana, A. Molfese, G. Marchesi, L. Atzeni, C. Buran, A. Paleari, G. Ballarin, S. Manzini, F. Alagi, G. Pizzo, "BCD8sP: An advanced 0.16 µm technology platform with state of the art power devices," IEEE International Symposium on Power Semiconductor Devices & IC's, 2013, pp. 361-364

[17] M. Grosso, S. Rinaudo, A. Casalino, M. Sonza Reorda, "Software-Based Self-Test for Transition Faults: a Case Study," IEEE International Conference on Very Large Scale Integration (VLSI-SoC), 2019, pp. 76-81