Test Chip Design for Optimal Cell-Aware Diagnosability*

Soumya Mittal, Zeye Liu, Ben Niewenhuis and R. D. Shawn Blanton
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA 15213
http://www.ece.cmu.edu/∼actl/

***Abstract*—Rapid yield learning in a new manufacturing process via test chips is greatly enhanced with a "Design for Diagnosis" methodology. Prior work on logic-based test chip design demonstrated an implementation flow that ensures 100% intra-cell fault coverage using a minimal test set. However, testability alone does not guarantee good diagnosability. Since diagnosis is inherently a function of design, it is crucial that the design flow ensures defect-level diagnosis resolution and accuracy. This work describes an enhanced implementation methodology for the Carnegie-Mellon Logic Characterization Vehicle (CM-LCV) that ensures optimal cell-aware diagnosability by design. Experiments comparing intra-cell defect diagnosability of the CM-LCV and various benchmark circuits demonstrate the efficacy.**

I. Introduction

Yield is always low when a new manufacturing process is introduced. Thus, in order to meet shrinking time-to-market and time-to-volume demands, the rate of yield ramping must be rapid. Different strategies are utilized for yield analysis depending on the maturity of the fabrication process. In-line inspection, test structures, and memories [1] are mostly used in the initial phases of yield learning. These structures however do not mimic all of the layout features inherently found within actual customer ICs, especially the logic circuits. Customer or product-like logic-based test chips, also known as logic characterization vehicles (LCVs) [2], are now commonly used by integrated device manufacturers, foundries and design houses. In addition to providing feedback about the IC manufacturing process, LCVs are designed and fabricated with more general goals such as evaluation of place-and-route and identification of troublesome layout patterns. However, the primary objective of an LCV remains to collect manufacturing feedback about a new technology node. Current approaches for LCV design focus on adapting existing product designs, resulting in non-optimal testability and diagnosability. A new type of product-like test chip called the Carnegie Mellon Logic Characterization Vehicle (CM-LCV) [3] optimizes test and diagnosis of test chips for yield improvement through careful design. The CM-LCV is a two-dimensional array of functional unit blocks (FUBs). The FUB array is designed to have guaranteed error propagation and constant testability [4], a property that ensures a fixed-size test set. Work in [5] describes a circular BIST architecture for the CM-LCV that achieves 100% fault coverage using the generalized input-pattern (IP) fault model [6] with a corresponding 86.9% test-time reduction. Other work in [7] describes a design flow that copies the physical characteristics of an example product into the CM-LCV. Finally, work in [8] discusses an approach to achieve 100% intra-cell fault testability while simultaneously ensuring the incorporation of product design features into the CM-LCV.

Although prior work discusses testability of interconnect and intra-cell defects of the CM-LCV in detail, diagnosability of the defects inside the standard cells and within the FUBs are not explicitly considered. Numerous publications exist in the literature for improving diagnosability of logic circuits, and essentially can be divided in two categories. One category focuses on developing diagnostic ATPG techniques [9]–[11]. These are used to enhance diagnosis resolution by generating tests that distinguish fault pairs. The other category centers on using improved diagnosis algorithms [12]–[16]. Other work (e.g., [17]–[20]) focuses specifically on providing better accuracy and resolution for cell-internal defects.

Besides better tests and algorithms, diagnosis also significantly depends on the design itself. "Design for Diagnosis" methods discussed in [21], [22] are based on the insertion of observation points but do not fundamentally alter the design methodology. An approach that creates the design from "scratch" with optimal defect localization accuracy and curtails physical failure analysis (PFA) time and effort would thus be very beneficial for constructing an effective LCV.

Because one of the challenges of fabricating a test chip on a new technology node is diagnosability [3], a design-for-diagnosis (DfD) technique is proposed in this work to overcome the drawbacks previously described. In this DfD technique, a matrix-based formulation is used to identify FUB implementations that ensures optimal cell-aware diagnosability. Several experiments are performed to evaluate the DfD methodology with results demonstrating significant diagnosis improvement for intra-cell defects within a FUB array as compared to benchmark circuits.

Prior work [8] achieved array testability by ensuring individual FUB testability. That is, optimal array testability is provably guaranteed through the use of constant-testability theory [4]. The same is not true however for diagnosability. In other words, making defects/faults inside the FUB and gates diagnosable

*This work is sponsored by the National Science Foundation under contract no. 1527606.

does not guarantee that those same defects/faults are diagnosable when the FUB is incorporated into an array of FUBs. This paper addresses this challenge. Specifically, (i) the FUB array design flow is modified to ensure each individual FUB by itself is diagnosable, and (ii) just as importantly, a method for guaranteeing array diagnosability is described when constructing a FUB array from individual diagnosable FUBs.

The rest of the paper is organized as follows. In Section 2, we briefly review relevant papers in logic diagnosis and define some of the terminology used throughout the paper. This section also details the need for a DfD methodology that includes experiments supporting this claim. Section 3 discusses the design flow and the optimization used to obtain the CM-LCV with improved cell-aware diagnosability. Experiment results assessing and validating the design flow are presented in Section 4. The final section concludes the paper and describes some directions for future work.

II. Background

The primary objective of diagnosis is to localize defects and if possible, identify their root cause. Diagnosis algorithms can be broadly categorized into cause-effect and effect-cause techniques. Cause-effect based approaches [23] build a database of simulated faulty behaviors and compare them with the observed tester responses. The database containing fault simulation responses is typically referred to as a fault dictionary. The quality of diagnosis significantly depends on the fidelity of the fault models used to build the dictionary. Conversely, effect-cause based approaches [24] analyze the faulty behavior and deduce one or more defects that adequately explains the observed behavior. A number of diagnosis algorithms have been described that follow either of these approaches, or employ a combination of these techniques to improve the outcome of diagnosis [12]–[16].

Additionally, diagnostic automatic test pattern generation (DATPG) methods enhance diagnostic resolution and accuracy by generating dedicated test patterns for distinguishing fault pairs [9]–[11]. For example, work in [9], [10] generates diagnostic test patterns to distinguish stuck-at faults while [11] considers arbitrary fault models derived from initial runs of diagnosis. However, these approaches can lead to a large number of test patterns and require significant runtime when the number of fault pairs is large. In general, the drawback for DATPG is its computational complexity due to extensive fault simulation and large test-set size. More importantly however, and often overlooked, is the fact that some faults are inherently indistinguishable due to the nature of the design itself.

To further improve diagnostic resolution, various techniques exist that report candidates at the transistor level instead of the gate level [17]–[20]. In [17], intra-cell faults are represented as gate-level faults so that conventional logic-based diagnosis approaches can be employed. In [18], excitation conditions are extracted for the cells identified from gate-level diagnosis tools and are matched with a fault dictionary created

TABLE 1
An example illustrating the IP fault model for a 2-input OR gate.

	Output Z				
Inputs A B	**Expected**	F_1	F_2	F_3	F_4
00	0	1	0	0	0
01	1	1	0	1	1
10	1	1	1	0	1
11	1	1	1	1	0

from switch-level simulation. Work presented in [19] demonstrates a methodology for fault characterization of standard-cell libraries using inductive contamination analysis. In [20], diagnosis is performed using analog fault models. Specifically, physical defects are identified from the standard-cell layout with corresponding fault models that are derived from analog simulation. Using these methods may not be suitable for diagnosing test chips since all possible defect mechanisms are not known a priori for a new fabrication process.

In order to be more general than a cell-aware fault model, we have adopted the Input Pattern (IP) fault model [6]. The IP fault model allows a circuit module to change its functionality arbitrarily, where a module can be arbitrarily defined (e.g., as a gate or some other sub-circuit). Table 1 illustrates how the IP fault model is applied to a simple gate. Specifically, it shows a truth table of an OR gate with inputs A and B, and output Z. The second column shows the correct function while the remaining columns capture the effect of the four possible IP faults. Note that the fault is defined as a change in a single row of the truth table. For a gate or a cell with n inputs and m outputs, there are $2^n(2^m - 1)$ different IP faults.

The IP fault model when exhaustively applied at the gate-level is the same as the gate-exhaustive fault model, that is, all IP faults are dominated by the $2^{m \cdot 2^n} - 1$ functional faults. Experiments in [25], [26] indicate that gate-exhaustive tests detect more defective chips than tests derived using conventional fault models.

To compare the diagnosability of different circuits, we use the metric defined in [10]. The metric, called Diagnostic Coverage (DC), is defined as

$$DC = \frac{g}{FT} \qquad (2.1)$$

where g is the number of detected fault groups and FT is the total number of faults. Here, a fault group consists of faults that produce the same output response for a given test set. These faults cannot be distinguished from each other for the given test set. Faults in different groups are differentiated from each other by the test set. Thus, the diagnostic coverage will be equal to one if every fault produces a unique response, implying ideal diagnostic resolution for defects that behave exactly like the faults.

Fig. 1 illustrates that fault coverage does not correlate very well with diagnosability. Specifically, it shows the distribution of more than 63,000 unique FUB implementations, and each plot point ("hexagon") denotes the number of FUB implementations with a given fault coverage and diagnosability value.

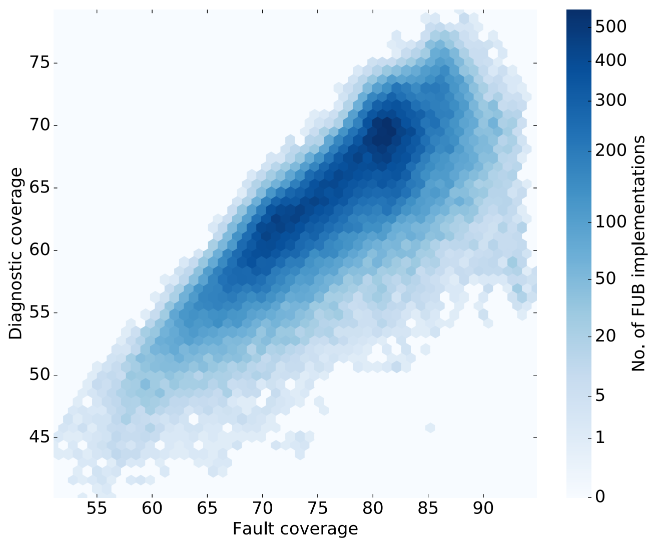
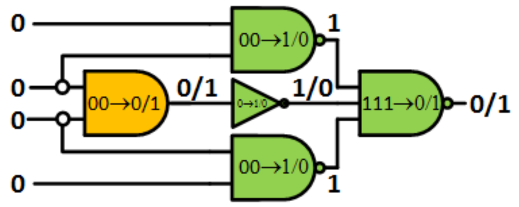
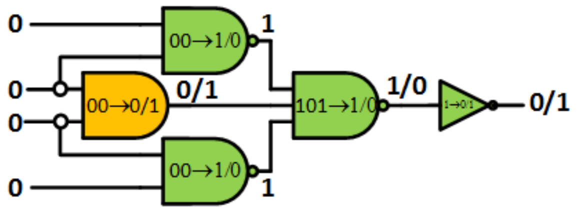


Figure 1. Plot comparing diagnostic coverage with fault coverage for over 63,000 FUB implementations.



(a)



(b)

Figure 2. Example illustrating the impact of design on diagnosis. For the circuit shown in (a), all the IP faults are detected which degrades resolution. For the slightly modified circuit of (b), fewer IP faults are detected for the same test pattern, thus improving resolution.

While it is apparent that increasing fault coverage does in general correlate with increasing diagnostic coverage, there is still significant variation in diagnostic coverage (roughly +/- 10%) for a given fault coverage. Thus, diagnostic coverage, as defined in Eq. 2.1, or some similar metric should be explicitly used to assess design diagnosability.

Because the fault effects of any detectable intra-cell defect can be captured by one or more IP faults, the IP fault model is utilized to calculate the diagnostic coverage of different circuits in order to capture their cell-aware diagnosability. A commercial standard-cell library is utilized to synthesize the circuits, and the fault simulator FATSIM [27] is used to simulate all cell-level IP faults.

Table 2 reports the outcome of this simulation, especially related to the redundant and missing IP faults. The last three columns of Table 2 reveal that there exists a number of IP fault classes [8] that are redundant and missing. An IP fault class is defined as a tuple consisting of a particular standard cell and a particular IP fault. An IP fault class is said to be redundant if the IP fault is redundant for each instance of its corresponding standard cell within a given design. The fourth column in Table 2 shows the number of redundant IP fault classes present in the design. The last two columns give the number of missing IP fault classes in the cell library, i.e., the number of classes corresponding to the logic functions and gates not included in the design. More specifically, the column labeled "Library" considers only standard cells with different logic functions while the column labeled "LibraryDS" also takes into account the driving strengths and functions of the standard cells.

TABLE 2
IP FAULT COVERAGE ANALYSIS FOR VARIOUS BENCHMARKS.

ISCAS89 benchmark circuit	No. of IP faults	No. of redundant IP faults	No. of redundant IP fault classes in circuit	No. of missing IP fault classes	
				Library	LibraryDS
s13207	8980	1354	22	86	1509
s15850	10864	1805	25	89	1515
s35932	23326	2117	1	205	1645
s38417	33770	2502	48	48	1444
s38584	42220	4896	27	27	1454

For the 154 standard cells in the library, there are 1,692 IP fault classes, of which 250 are independent of drive strengths. From Table 2, we notice that a significant percentage of IP fault classes are missing or redundant for the benchmark designs analyzed. This is not favorable for yield learning; for example, a systematic defect in a cell cannot by observed if all of its IP fault classes are either redundant or missing. An ideal test chip should include all IP fault classes, and none of them should be redundant.

As mentioned, diagnosis is either improved (typically) by using advanced algorithms and/or by using higher quality test patterns. But as already mentioned, another important factor that influences diagnosis is the design itself. Attempts have been made in the direction of "Design for Diagnosis" (e.g. [21] and [22]) but they are essentially based on the insertion of observation points instead of the design implementation itself.

The circuit in Fig. 2 illustrates how design can influence diagnosis. It shows the fault-effect propagation of an IP fault for the 2-input AND gate. We notice that for the only test pattern that activates the AND IP fault (00→0/1), the fault effects (not shown) from the two 2-input NAND gates are also propagated to the output for circuit of Fig. 2a. However, this does not occur for the slightly altered circuit of Fig. 2b. In fact, the NAND IP faults shown are now redundant for the circuit of Fig. 2b. This example therefore illustrates that even a slight change in design topology can change both the diagnostic resolution and fault redundancy.

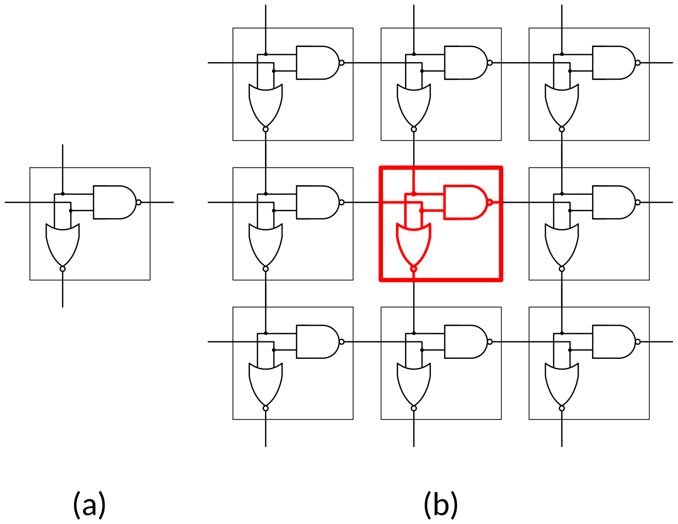


Figure 3. Example illustrating how diagnostic coverage decreases when a FUB is within an array.

One important question that must be addressed is the relationship between FUB-level resolution and array-level resolution. Specifically, what can be said about the diagnostic coverage of a FUB array (DC_{array}) composed of individual FUBs (FB_1, FB_2, ..., FB_n), each with a corresponding DC_i (DC_1, DC_2, ..., DC_n). In the ideal situation, where no cell-level IP fault from any FUB FB_i is equivalent to any other IP fault from any other FUB FB_j ($i \neq j$), the DC_{array} would be a simple composition. That is, $DC_{array} = \sum g_i / \sum FT_i$, where each g_i is the number of fault groups for FB_i, and FT_i is the total number of cell-level IP faults in FB_i. This ideal case is desirable because it means FUBs can be selected based on their diagnosability characteristics independent from other FUBs in the array, regardless of their location.

To illustrate how diagnostic coverage can decrease when FUBs are arranged in an array, consider the circuits shown in Fig. 3. Fig. 3a shows a simple 2-input, 2-output FUB. For an exhaustive set of test patterns, we observe that each IP fault in the example FUB has a unique fault signature. This means the FUB exhibits ideal diagnostic resolution in isolation, that is, when the FUB by itself is viewed as the circuit under test, diagnostic resolution is ideal. Now, when this FUB is placed within a 3×3 array, as shown in Fig. 3b, and tested with an exhaustive, array-level test set, we observe that the resolution of some of the IP faults for the highlighted FUB degrades. Specifically, Table 3 lists the resolution (the number of candidates reported by diagnosis) of all the FUB IP faults when isolated, and after being placed within the array. Two IP faults in the highlighted FUB of Fig. 3, F_2 and F_6, become redundant due to the controllability and observability constraints imposed by the array. Furthermore, the resolution of two other faults, F_4 and F_5, degrades due to fault equivalence among faults within different FUBs in the array (i.e. inter-FUB fault equivalence). Thus, this simple example illustrates how diagnosability of FUBs in isolation is not necessarily maintained when the FUBs are within an array.

TABLE 3
COMPARISON OF DIAGNOSTIC RESOLUTION OF THE FUB AND FUB ARRAY SHOWN IN FIG. 3.

Fault index	IP Fault	Module-level resolution (Fig. 3a)	Array-level resolution (Fig. 3b)
F_1	NAND2, 00→1/0	1	1
F_2	NAND2, 01→1/0	1	Redundant
F_3	NAND2, 10→1/0	1	1
F_4	NAND2, 11→0/1	1	4
F_5	NOR2, 00→1/0	1	4
F_6	NOR2, 01→0/1	1	Redundant
F_7	NOR2, 10→0/1	1	1
F_8	NOR2, 11→0/1	1	1

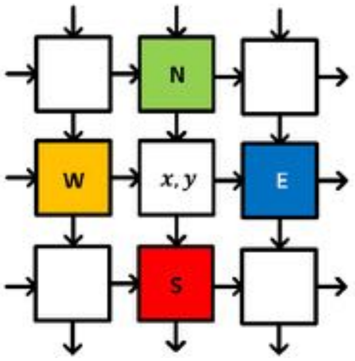


Figure 4. Faults within the (x, y)-FUB can only possibly be equivalent to faults within the driving (north or west) or driven (south or east) neighbors.

Analysis of the fault-detection properties of a two-dimensional array of VH-bijective[1] FUBs [3] has revealed that only neighboring FUBs can possess defects/faults that are equivalent. This means that a FUB located at position (x,y) can only have inter-FUB defect/fault equivalencies with the FUBs located immediately to the north, west, east or south, as shown in Fig. 4. Moreover, the necessary conditions for two inter-FUB faults to be equivalent are quite difficult to satisfy. There are two possible cases that must be examined.

For the first case, assume the (x,y)-FUB of Fig. 4 produces an error at both the horizontal and vertical outputs. For this case, it is not at all possible for the driven FUBs (locations E and S) to possess a defect/fault that would mimic the error propagation ensured by the function implemented within each VH-bijective FUB. It is possible however for either of the two FUBs driving the (x,y)-FUB (locations N and W) to possess a defect/fault that produces the same array response. These potentially-equivalent faults in adjacent FUBs can be derived or proven not to exist for each FUB fault of concern.

For the second case, assume the (x,y)-FUB of Fig. 4 produces an error exclusively at either its horizontal or vertical output. In this case, it is not possible for the driving FUBs (locations N and W) to possess an equivalent defect/fault, as above. However, the driven FUBs (locations E and S) can

[1]A VH-bijective function ensures propagation of one or more errors present at either a horizontal or vertical input to at least one vertical and horizontal output.

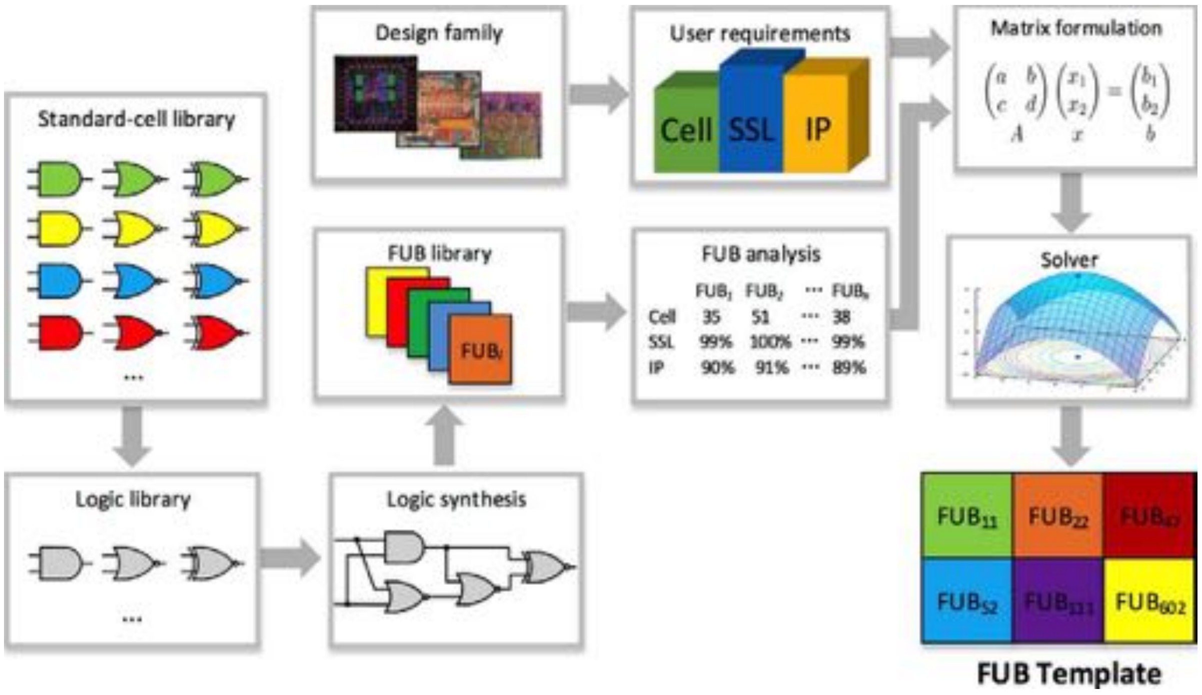


Figure 5. The CM-LCV implementation flow.

possess an equivalent defect/fault. Again, these potentially-equivalent faults in adjacent FUBs can be derived or proven not to exist for each FUB fault of concern.

In order for an array to have inter-FUB fault equivalencies, there must exist two neighboring FUBs, each with a fault signature[2] that perfectly matches this constructed equivalent fault for the other. Thus, if an array is restricted to FUBs with no possible equivalent faults, then inter-FUB fault equivalences cannot exist. Moreover, no inter-FUB fault equivalences can exist if only one of the equivalent faults (from the driving or driven FUB) exist since both obviously must be present to establish the equivalency.

In a preliminary analysis, over eight thousand FUB implementations are examined, and only 2.6% possessed cell-level IP faults that meet the necessary conditions for inter-FUB fault equivalence. Hence, by eliminating these FUB implementations from the FUB library, it is possible to ensure ideal diagnostic coverage composition among all of the FUBs in the library. Thus, focusing on diagnosability of individual FUBs will ensure that the diagnosability of the FUB array will coincide with the ideal case described earlier.

III. Design for Diagnosis

This work builds on the LCV design methodology used in [8] where only intra-cell testability is considered. Specifically, the methodology in [8] synthesizes a large number of FUB implementations to guarantee that no IP fault class is redundant or missing. Moreover, it incorporates the physical characteristics of a product design into the CM-LCV. Here this design flow is significantly enhanced to guarantee cell-aware diagnosability through design.

[2]A fault signature is the simulation response produced by a single FUB affected by a fault (e.g., an IP fault) when it is exhaustively tested.

Fig. 5 illustrates the design-for-diagnosis (DfD) methodology. The first step creates a library that only contains logic functions extracted from the standard-cell library. The cell functions usually come in low power and multiple drive strength variants; but standard cells with the same function but different physical layouts are not distinguished in this step of the flow. The second step is synthesis; this step generates a large number of FUB implementations using different subsets of standard-cell functions from the logic library. All of these FUB implementations taken together form the FUB library. Each FUB implementation in this library realizes the same bijective function but utilizes different cells. The third step characterizes each FUB in the library in the following way:

1) The vector **C** stores the number of instances of each library-function type used within the FUB module. Specifically, each component $\mathbf{C}[l] \in \mathbf{C}$ is equal to the number of times the cell function l is instantiated in the FUB module.
2) The vector **R** tracks the number of redundant IP faults for each IP fault class. Specifically, each component $\mathbf{R}[(l,\ f_k)] \in \mathbf{R}$ is equal to the number of redundant IP faults f_k for IP fault k and cell function l.
3) The vector **FC** combines vectors **C** and **R** resulting in a measure of IP fault class coverage. Specifically, **FC** is:
   - $\mathbf{FC}[(l, f_k)] = 1 \leftrightarrow \mathbf{C}[l] > 0$ and $\mathbf{C}[l] > \mathbf{R}[(l, f_k)]$
   - $\mathbf{FC}[(l, f_k)] = 0$, otherwise.

   This means that an IP fault class for a FUB is considered covered when at least one IP fault class for the gate in the FUB is tested.

4) The vector **DC** represents the diagnostic coverage of each FUB implementation calculated using Eq. 2.1. Specifically, each component $\mathbf{DC}[i] \in \mathbf{DC}$ is the diagnostic coverage for FUB implementation i. The vector **DR** is calculated by subtracting $\mathbf{DC}[i]$ of each FUB implementation from one. Specifically, $\mathbf{DR}[i] = 1 - \mathbf{DC}[i]$.

The next step in the flow combines the vector information for each FUB to form a matrix that is solved using an optimization solver. The output of the solver is a FUB *template*, i.e., a set of FUB implementations from the library that achieves optimum diagnosis coverage without any redundant or missing IP fault classes. The optimization performed by the solver can be expressed as:

$$\begin{aligned} &Minimize\ (\frac{DRx}{x}) \\ &Subject\ to\quad FCx \geq d \\ &\qquad\qquad x \geq 0 \end{aligned} \tag{3.1}$$

where d is a matrix that guarantees each IP fault class is testable, DR and FC are as defined above, and x is the solution that represents the set of FUB implementations identified by the solver, i.e. the FUB template.

The second part of Fig. 5 derives standard-cell demographics (i.e., cell instance counts) from a design family. Alternatively, cell demographics can be specified by the user. In addition to achieving high diagnostic resolution, it is also important to mimic or reflect the physical characteristics of a design. In this work, the reflection of standard-cell demographics is considered a good approximation of the critical design properties. The following equation optimizes diagnostic coverage while incorporating cell demographics into the CM-LCV.

$$\begin{aligned} &Minimize\ (j(\frac{DRx}{x}) + \|Cx - b\|^2) \\ &Subject\ to\quad FCx \geq T \\ &\qquad\qquad x \geq 0 \end{aligned} \tag{3.2}$$

where b is a matrix that represents the targeted demographics or user-specified design requirements, T is a constraint matrix that guarantees that a certain number of IP faults for a given fault class are diagnosable within the FUB template, j is a tuning parameter, and C, DR, FC and x have the same meaning as described earlier.

Equation 3.1 attempts to maximize the diagnostic coverage of the FUB implementations in the resulting FUB template. In addition to maximizing the diagnosability of the FUB template, Equation 3.2 also tries to minimize the differences in standard-cell demographics between the template and the targeted demographics. The constraints used in both equations ensure that no IP fault class will be redundant or missing.

IV. Experiment

For the experiments described here, the standard-cell library that is utilized to synthesize different benchmark circuits in Section 2 is also used here. Following the design flow detailed in Section 3, a FUB library is created that contains more than 63,000 different FUB implementations of a VH-bijective function that has a total of six inputs and six outputs, equally split between the vertical and horizontal ports. The logic cells used in these FUB implementations are replaced with corresponding standard-cell functions, with their driving strengths assigned randomly.

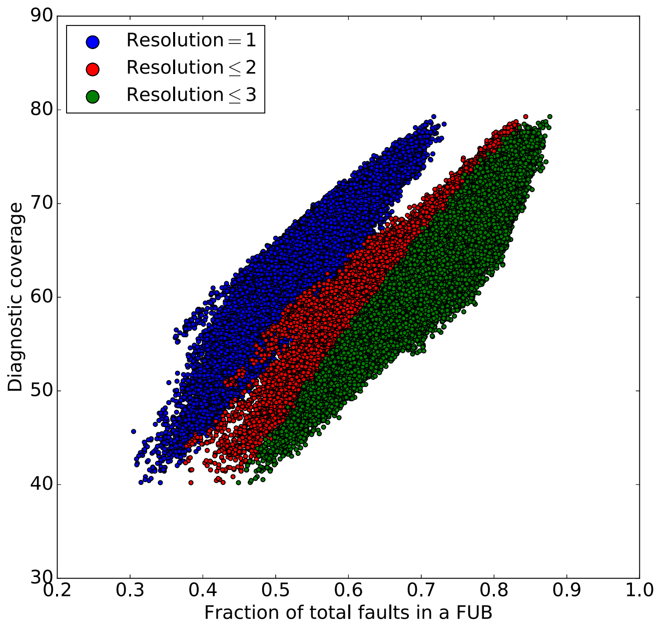


Figure 6. Distribution of diagnostic coverage for over 63,000 FUB implementations for each resolution criterion. Each point corresponds to a FUB with y-axis value being its diagnostic coverage and x-axis value being the fraction of IP faults having a particular resolution value or range of values (denoted by different colors).

Fig. 6 illustrates how diagnostic coverage correlates with individual fault resolution within a given FUB implementation. Each plot point corresponds to a unique FUB implementation of a FUB function that guarantees optimal testability of a FUB array, and there are over 63,000 points per color plotted. Consider the distribution of points shown in blue. For a FUB located at coordinate x-y in this distribution, the y-value denotes its diagnostic coverage, and the x-value is the fraction of IP faults in the FUB with ideal resolution. Similarly, for the distribution of points shown in red (green), the x-value is the fraction of faults with resolution less than or equal to two (three) and the y-value again is the diagnostic coverage. It is observed that for FUB implementations with high diagnostic coverage, around 90% of the faults have resolution less than or equal to three, out of which 70% of faults have ideal resolution. Thus, selecting implementations with high DC for realistic and/or comprehensive fault models such as the IP fault model greatly increases the likelihood of uncovering the root-cause of failure for a defective FUB.

After characterizing the FUB library and removing those FUB implementations with potential inter-FUB fault equivalencies, Equation 3.1 is solved using the criteria discussed in

TABLE 4
DIAGNOSTIC COVERAGE COMPARISON FOR A FUB TEMPLATE AGAINST VARIOUS BENCHMARKS USING THE IP AND SSL FAULT MODELS.

Circuit	No. of missing IP fault classes	IP DC	Percentage of redundant SSL faults	SSL DC
s13207	1509	47.41	15.65	47.04
s15850	1515	57.06	18.08	46.47
s35932	1645	81.45	2.05	75.89
s38417	1444	59.91	21.42	46.31
s38584	1454	69.96	10.12	57.36
Family	982	66.48	11.84	56.95
FUB$_t$	0	72.53	0	83.11

Section 3. The solver, Branch and Reduce Optimization Navigator (BARON) [28], is used to obtain the FUB template. Table 4 shows a detailed comparison of the diagnostic coverage of the FUB template with full-scan ISCAS89 benchmark circuits for both the IP and stuck-at (SSL) fault models. The number of missing IP fault classes and the percentage of redundant SSL faults are reported in second and fourth columns, respectively. The diagnostic coverage for both the IP and SSL fault models are listed in third and fifth columns, respectively. The first five rows report results of the benchmark circuits, with all five benchmark circuits evaluated together as a design family in the sixth row. The last row reports results for the FUB template (FUB$_t$). It is evident from the table that the FUB template achieves higher diagnostic coverage than all of the benchmark circuits except s35932. However, note that the number of missing IP fault classes in this benchmark is significant, i.e., many standard cells are unused while the FUB template does not have any missing or redundant IP fault classes. Though s35932 has better diagnosability than the FUB template for the IP fault classes it contains, it has a significant number of missing IP fault classes making it a very poor candidate for a test chip. Additionally, the FUB delivers better diagnosability when the SSL fault model is considered.

Besides attaining high diagnosis resolution without any redundant or missing IP fault classes, incorporating standard-cell demographics into the CM-LCV FUB array is desirable [7]. The FUB templates, for both cell reflection and diagnosability, resulting from solving Equation 3.2 are tabulated in Table 5 for the IP fault model, and in Table 6 for the SSL fault model. The design characteristics of the ISCAS design family are incorporated into two different FUB templates. We observe that there is a trade-off between achieving high diagnostic coverage and low cell-demographics mismatch. Cell-demographics mismatch is defined as the sum of absolute differences of the number of logic cells of each function type between the design and the FUB template, divided by the total number of cell instances utilized in the design. Higher diagnosability can be achieved with a larger value for the scalar multiplier j in Equation 3.2, whereas low demographics mismatch can be reached with a comparatively smaller j value. The optimal trade-off between these two properties, i.e. diagnosability and design similarity (as measured by cell-demographics mismatch) is difficult to predict due to the complex relationship between the design, the manufacturing process, and the defect mechanisms.

TABLE 5
TRADE-OFF BETWEEN DIAGNOSABILITY AND CELL REFLECTION FOR THE IP FAULT MODEL.

Circuit	IP DC	Cell-demographics mismatch (%)	Tuning parameter
FUB$_{ip1}$	63.11	34.39	7000
FUB$_{ip2}$	70.61	67.49	100000

TABLE 6
TRADE-OFF BETWEEN DIAGNOSABILITY AND CELL REFLECTION FOR THE SSL FAULT MODEL.

Circuit	SSL DC	Cell-demographics mismatch (%)	Tuning parameter
FUB$_{ssl1}$	72.52	33.02	1000
FUB$_{ssl2}$	77.31	44.54	3000

TABLE 7
DIAGNOSTIC COVERAGE COMPARISON FOR A FUB TEMPLATE AGAINST ISCAS DESIGN FAMILY FOR DIFFERENT FAULT MODELS.

Circuit	SSL DC	IP DC	SLIDER DC
FUB$_t$	83.11	72.53	73.42
ISCAS	56.95	66.48	67.43

TABLE 8
TRADE-OFF BETWEEN DIAGNOSABILITY AND CELL REFLECTION FOR INTRA-CELL DEFECTS DERIVED USING INDUCTIVE-FAULT ANALYSIS.

Circuit	SLIDER DC	Cell-demographics mismatch (%)	Tuning parameter
FUB$_{slider1}$	65.69	34.53	10000
FUB$_{slider2}$	67.89	41.09	100000

To further analyze intra-cell diagnosability, SLIDER [29] is used to extract the most-likely intra-cell defect behaviors. (SLIDER is a defect simulation framework and is used here to model realistic intra-cell defects.) Accordingly, only the IP faults that capture the intra-cell defects identified by SLIDER are considered. Table 7 compares the diagnostic coverage of ISCAS designs and the FUB template obtained from solving Equation 3.1 for different fault models and SLIDER-identified defects. We observe that the intra-cell diagnosability based on the SLIDER-identified defects is better than IP fault diagnosability. Furthermore, Table 8 shows the FUB templates for induced cell-aware defects from the standard-cell layouts when design reflection is considered and produces similar results. It is therefore possible to achieve better demographics mismatch at the cost of diagnostic coverage when also performing inductive fault analysis.

V. CONCLUSION

In this paper, a Design-for-Diagnosis technique is described that improves the diagnosability of logic-based test chip while minimizing the number of redundant and missing IP fault classes. Specifically, a methodology is developed for the CM-LCV to ensure optimal cell-aware diagnosability by design.

The experimental results indeed show better diagnosability for the CM-LCVs produced using this methodology compared to the benchmark circuits. Moreover, it is shown that the proposed methodology can optimize the diagnostic resolution while incorporating physical characteristics of a product design in the CM-LCV and ensuring that all IP fault classes are tested. Our future work will be focused on improving the diagnosability of multiple defects and incorporating specific layout features into the LCV. Finally, we are in the process of taping-out an LCV in volume with an industrial partner in 7nm technology.

References

[1] D. J. Ciplickas, X. Li, and A. J. Strojwas, "Predictive Yield Modeling of VLSICs," *International Workshop on Statistical Metrology*, pp. 28–37, 2000.

[2] C. Hess *et al.*, "Logic Characterization Vehicle to Determine Process Variation Impact on Yield and Performance of Digital Circuits," *International Conference on Microelectronic Test Structures*, pp. 189–196, 2002.

[3] R. D. Blanton, B. Niewenhuis, and C. Taylor, "Logic Characterization Vehicle Design for Maximal Information Extraction for Yield Learning," *IEEE International Test Conference*, Oct. 2014.

[4] A. D. Friedman, "Easily Testable Iterative Systems," *IEEE Transactions on Computers*, vol. C-22, no. 12, pp. 1061–1064, Dec. 1973.

[5] B. Niewenhuis and R. D. Blanton, "Efficient Built-in Self Test of Regular Logic Characterization Vehicles," *IEEE VLSI Test Symposium*, 2015.

[6] R. D. Blanton and J. P. Hayes, "Properties of The Input Pattern Fault Model," *IEEE International Conference on Computer Design*, Oct. 1997.

[7] R. D. Blanton, B. Niewenhuis, and Z. Liu, "Design Reflection for Optimal Test-Chip Implementation," *IEEE International Test Conference*, Oct. 2015.

[8] Z. Liu *et al.*, "Achieving 100% Cell-Aware Coverage by Design," *Design, Automation and Test in Europe*, 2016.

[9] A. Veneris *et al.*, "Fault equivalence and diagnostic test generation using ATPG," *Proceedings of the International Symposium on Circuits and Systems*, pp. 221–224, 2004.

[10] Y. Zhang and V. D. Agrawal, "A Diagnostic Test Generation System," *IEEE International Test Conference*, pp. 1–9, 2010.

[11] N. K. Bhatti and R. D. Blanton, "Diagnostic Test Generation for Arbitrary Faults," *IEEE International Test Conference*, 2006.

[12] S. Venkataraman and S. B. Drummonds, "POIROT: A Logic Fault Diagnosis Tool and Its Applications," *Proceedings International Test Conference*, pp. 253 – 262, 2000.

[13] L. M. Huisman, "Diagnosing Arbitrary Defects in Logic Designs Using Single Location at a Time (SLAT)," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 1, pp. 91–101, 2004.

[14] R. Desineni, O. Poku, and R. D. Blanton, "A Logic Diagnosis Methodology for Improved Localization and Extraction of Accurate Defect Behavior," *IEEE International Test Conference*, pp. 1–10, 2006.

[15] X. Yu and R. D. Blanton, "Improving Diagnosis through Failing Behavior Identification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 10, pp. 1614–1625, 2012.

[16] A. Riefert *et al.*, "Improving Diagnosis Resolution of a Fault Detection Test Set," *VLSI Test Symposium*, pp. 1–6, 2015.

[17] X. Fan *et al.*, "Extending Gate-Level Diagnosis Tools to CMOS Intra-Gate Faults," *Proceedings of IET Computer & Digital Techniques*, vol. 1, no. 6, pp. 685–693, 2007.

[18] E. Amyeen, D. Nayak, and S. Venkataraman, "Improving Precision Using Mixed-level Fault Diagnosis," *IEEE International Test Conference*, pp. 318–323, 2006.

[19] J. Khare, W. Maly, and N. Tiday, "Fault Characterization of Standard Cell Libraries Using Inductive Contamination Analysis (ICA)," *IEEE VLSI Test Symposium*, pp. 405–413, May 1996.

[20] H. Tang *et al.*, "Diagnosing Cell Internal Defects Using Analog Simulation-based Fault Models," *Asian Test Symposium*, pp. 318–323, 2014.

[21] I. Pomeranz, S. Venkataraman, and S. M. Reddy, "Z-DFD: Design-for-diagnosability based on the concept of Z-detection," *Proceedings International Test Conference*, pp. 489–497, 2004.

[22] Z. Li *et al.*, "Efficient Observation-Point Insertion for Diagnosability Enhancement in Digital Circuits," *IEEE International Test Conference*, Oct. 2015.

[23] J. Richman and K. R. Bowden, "The Modern Fault Dictionary," *Proceedings International Test Conference*, pp. 696–702, 1985.

[24] H. Cox and J. Rajski, "A Method of Fault Analysis for Test Generation and Fault Diagnosis," *IEEE Transactions on Computer-Aided Design*, 1988.

[25] K. Cho, S. Mitra, and E. McCluskey, "Gate Exhaustive Testing," *IEEE International Test Conference*, Nov. 2005.

[26] R. Guo *et al.*, "Evaluation of Test Metrics: Stuck-at, Bridge Coverage Estimate and Gate Exhaustive," *Proceedings VLSI Test Symposium*, pp. 66–71, 2006.

[27] K. N. Dwarakanath and R. D. Blanton, "Universal Fault Simulation Using Fault Tuples," *Proceedings Design Automation Conference*, pp. 786–789, 2000.

[28] Sahinidis, V. Nikolaos, and M. Tawarmalani, "BARON: Global Optimization of Mixed-integer Nonlinear Programs." User's manual, 2005.

[29] W. C. Tam and R. D. Blanton, "SLIDER: Simulation of Layout-Injected Defects for Electrical Responses," *IEEE Transactions on Computer-Aided Design*, vol. 31, no. 6, pp. 918–929, June 2012.