Logic Characterization Vehicle Design for Maximal Information Extraction for Yield Learning

R. D. (Shawn) Blanton, Ben Niewenhuis and Carl Taylor
Advanced Chip Testing Laboratory
ECE Department
Carnegie Mellon University

Abstract
A new type of logic characterization vehicle (LCV) that optimizes design, test, and diagnosis for yield learning is described. The Carnegie-Mellon LCV (CM-LCV) uses constant-testability theory and logic/layout regularity to create a parameterized design that exhibits both front- and back-end characteristics of a product-like, customer design. Design and test analysis of various CM-LCV designs (one of which has >4M gates) demonstrates that design time and density, test and diagnosis can all be simultaneously improved. For example, conventional ATPG produces test sets that are 2X larger, produce significantly poorer fault-detection and diagnostic characteristics for standard and advanced fault models, and require runtimes that are several orders of magnitude larger than the simple approach used to generate the constant test set for the CM-LCV. On the design side, a fully designed custom layout is 25% smaller than its synthesized, place-and-routed counterpart.

1 Introduction
It is now common practice for design houses (e.g., Nvidia and Qualcomm), IC fabricators (e.g., Global Foundries and TSMC), and integrated device manufacturers (e.g., Samsung and Intel) to fabricate product-like test chips that have functional characteristics similar to customer products. These test chips are not meant to be sold as products but are instead example ICs that provide feedback about the design and fabrication methodology and technology. Because test chips are not sold for profit, their volume is typically low in order to minimize cost, and their design is currently ad hoc in nature in that they are typically composed of smaller portions of existing or past designs that have been scaled to the current technology node. Moreover, such designs are not optimal in that they are not ideal vehicles for providing design and fabrication feedback. A survey that we conducted of various foundries and fabless companies revealed that the feedback sought from product-like test chips is quite varied as shown in Table 1. The same survey (Table 2) also revealed that there is an equal variety of important challenges that need to be addressed.

This paper describes a design and implementation methodology for the logic portion of a product-like test chip which from now on will be referred to as a logic characterization vehicle (LCV). Specifically, the approach is hierarchical in nature since the logic design (i.e., RTL, netlist, etc.) of the LCV can be somewhat treated independently with respect to its physical design (i.e., the layout). This means that test and diagnosis of the logic can be optimized while independently exploring a variety of physical design and fabrication styles (e.g., double- and triple-patterning [1], pattern regularity [2], standard cell channel routing [3], dummy fill [4], etc.). At the logic level, we exploit constant-testability (C-testability) [5], a property that certain logic circuits can be designed to possess that enables efficient pseudo-exhaustive test of all sub-circuits with a constant, small number of test patterns. In addition, the LCV includes periphery circuitry that enables both efficient on-chip test-pattern application of the constant test set for the CM-LCV. Finally, the LCV also exhibits extreme density which means a low-wafer volume will produce a statistically-significant sample of design-fabrication-test interactions.
P&R Place-and-route (P&R) has a significant impact on performance and yield, thus any feedback concerning the various styles of P&R and their corresponding impact on performance, reliability, and yield are greatly sought.

BEOL DFM Design-for-manufacturability guidelines are recommended constraints on the layout that are meant to improve yield. But the relative importance of these constraints is typically not well understood for a given design, especially for those guidelines aimed at the metal interconnect fabricated at the back-end of the fabrication line (BEOL). Thus any feedback concerning relative importance of the large variety of DFM guidelines is greatly sought.

Systematics The ability to find known yield detractors (e.g., layout geometries that do not print well, dummy fill styles that do polish well, etc.) using test chips is of critical importance and one of the main drivers for designing and fabricating test chips.

FEOL Many fabless companies are now beginning to design their own standard-cell libraries. As a result, cell evaluation is becoming a critical task that is tackled using test chips.

Table 1: Survey feedback concerning the objectives for product-like test chips.

<table>
<thead>
<tr>
<th>Feedback</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P&amp;R</td>
<td>Place-and-route (P&amp;R) has a significant impact on performance and yield, thus any feedback concerning the various styles of P&amp;R and their corresponding impact on performance, reliability, and yield are greatly sought.</td>
</tr>
<tr>
<td>BEOL DFM</td>
<td>Design-for-manufacturability guidelines are recommended constraints on the layout that are meant to improve yield. But the relative importance of these constraints is typically not well understood for a given design, especially for those guidelines aimed at the metal interconnect fabricated at the back-end of the fabrication line (BEOL). Thus any feedback concerning relative importance of the large variety of DFM guidelines is greatly sought.</td>
</tr>
<tr>
<td>Systematics</td>
<td>The ability to find known yield detractors (e.g., layout geometries that do not print well, dummy fill styles that do polish well, etc.) using test chips is of critical importance and one of the main drivers for designing and fabricating test chips.</td>
</tr>
<tr>
<td>FEOL</td>
<td>Many fabless companies are now beginning to design their own standard-cell libraries. As a result, cell evaluation is becoming a critical task that is tackled using test chips.</td>
</tr>
</tbody>
</table>

Table 2: Top challenges for using product-like test chips for yield learning.

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Test is the primary method for receiving feedback from a product-like test chip. It is therefore of the utmost importance to thoroughly test each fabricated instance. Thus, the challenge here is the generation of test patterns that not only detect failure but also support root-cause characterization.</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>Test is only the first step in characterizing failure. Follow-on diagnosis is also needed to localize the failure not only at the logical level but the physical level. Ideally, the combination of test and diagnosis should pinpoint the exact location and cause of failure.</td>
</tr>
<tr>
<td>Cost</td>
<td>Cost is also a challenge. Because test chips are not sold, the design, silicon and test resources aimed at test chips cannot be directly offset by product profit. There is then significant pressure to extract as much statistically-significant data from a small volume of logic test chips.</td>
</tr>
</tbody>
</table>

The rest of this paper describes the details of our approach for the design, test and diagnosis of a specialized logic characterization vehicle that we call CM-LCV (Carnegie-Mellon Logic Characterization Vehicle). Specifically, section 2 provides a brief overview of classical test structures used for fabrication characterization along with a description of other known LCVs. There we will also contrast the properties of existing LCV designs with the CM-LCV. Next, in Section 3, the design and test of the CM-LCV is presented in detail, while Section 4 presents various test and diagnosis experiments that demonstrates the advantages of the CM-LCV. Finally, conclusions and current work are given in Section 5.

2 Background
Because no one approach can effectively find and/or prevent every manufacturing perturbation, the semiconductor industry employs various design- and fabrication-related procedures to maintain and improve yield. At the design level for example, manufacturers impose constraints (i.e., rules and guidelines) that either forbids or discourages the use of certain layout geometries since there is ample evidence they fail at an unacceptable rate. On the manufacturing side, a plethora of yield-learning techniques involving in-line inspection, scribe-line structures of various types, and physical failure analysis of actual IC failures that appear in significantly-sized paretoes are all employed to identify shortcomings in the fabrication process, the design, or some combination thereof [6]. Consider test structures for example. Test structures [7] can range from being very simple in nature (combs and serpentine interconnect for measuring defect density size distributions (DDSDs) [8,9]) to more complex, active structures that includes transistors, ring oscillators, and SRAMs. Test structures are designed to provide seamless access for a particular type of failure, it size, its location, and possibly other pertinent characteristics. For example, Figure 1 shows a comb structure for measuring DDSDs for bridge defects. A contaminating particle that conducts and has a diameter equal to or greater than \( d \) will bridge the two conductors \( A \) and \( B \). An impedance measurement taken between the conductors reveals the presence of a defect. By
adjusting \( d \) and the number of identical copies of the comb structure, the DDSD can be adequately sampled. But the downside of combs and other test structures is that they cannot be sold for profit, hence they are typically relegated to the scribe lines (or a limited number of wafer lots), and are small in size. These two characteristics of test structures are also a source of major disadvantage. Specifically, since test structures are typically small, highly regular, and utilize a subset of the available fabrication layers, they cannot mimic the layout diversity typically found in customer ICs. Also, since they are limited to a small number of wafer lots or the scribe lines, their sample size of the fabrication process is also limited.

Figure 1: Comb structure for measuring defect density and size distributions.

Using customer ICs for yield learning seems to be a very natural choice since it is the targeted recipient of yield-learning benefits. One of the main challenges in accomplishing this however has been the problem of de-convoluting the IC tester response for not only failure localization at the logical (“netlist”) level (which is the conventional objective of diagnosis) but also for defect characterization which involves deriving the defect type, its size, its three-dimensional physical location in the layout, and ideally its root cause. Researchers for a number of years have explored the use of test results from the actual product IC for yield learning [10-13]. This is not a totally new idea however in that others have in the past proposed the use of customer IC test data to extract valuable information about the IC fabrication process (e.g., [14-16]).

A logic characterization vehicle (LCV) stands between the classical test structure and the customer IC. Like the classic test structure, an LCV is designed and fabricated for the sole purposes of providing feedback about design and fabrication. Like the customer IC, an LCV can employ both DRC and DFM to create cell-level netlists that are place-and-routed using commercially-available design flows. For example, the authors in [17] describe an LCV that consists of a logic circuit that is surrounded by a scan-based “JIG”, which is surrounding circuitry that enables scan and performance-based testing of the combinational circuit. Performance testing is accomplished through a configurable ring oscillator (RO). Specifically, the RO is established under the control of the JIG by making either an inverting or non-inverting connection from a circuit output back to one of its inputs. The JIG uses a counter and a special start-and-stop procedure to make a glitch-free performance measurement.

In [18], the authors describe an LCV consisting of an array of flip-flops with SRAM-like connections (bit lines, word lines, etc.). By utilizing a memory-like I/O scheme, the authors can apply memory test techniques to their LCV. Additionally, they argue that the flip-flop array can be made to reflect a typical logic layout through synthesis with a standard-cell library, and the use of conventional place-and-route flows. Finally, product-like BEOL routes are mimicked by having multiple paths for the bit-lines/word-lines that are individually selected using multiplexers.

The CM-LCV is completely compatible with the LCV capabilities described in [17] and [18]. In addition, as detailed in the following sections, the CM-LCV is optimized for test and diagnosis. The characteristics that make the CM-LCV highly testable and diagnosable also lend themselves to a physical design that is simultaneously dense (i.e., maximizes area utilization) and parameterizable (i.e., can be easily sized to fit any footprint without loss in density). This latter property is extremely advantageous since LCV volume (i.e., the number of fabricated instances) is extremely limited due to cost.

3 Design and Test Methodology

Details of the design and test properties/procedures of the CM-LCV are described here. Test properties are described first since they enable the optimized logical/physical design properties in Section 3.2.

3.1 LCV Test

Regular logic circuits are composed of identical functional unit blocks (FUBs) that are interconnected in a uniform fashion and include one-dimensional arrays, convergent and divergent trees, and two-dimensional arrays. Various types of data-path circuits, including adders, multipliers, parity circuits, multiplexers, and decoders, have regular circuits as
major components. In 1973, Friedman described the necessary and sufficient conditions for one-dimensional array circuits (without vertical outputs) to be what he called C-testable [5]. A regular logic circuit is C-testable if every FUB-level input pattern (IP) fault [19] can be tested with a constant number of test patterns, independent of the size of the circuit. In other words, a C-testable circuit implies that the function of every circuit FUB can be exhaustively verified with a fixed number of test patterns. Later, necessary and sufficient C-testability conditions for one-dimensional arrays with vertical outputs [20] and tree circuits [21] have been derived. But necessary and sufficient conditions for two-dimensional array C-testability remain an open problem despite extensive studies [e.g., 22-25].

In addition to the number of tests being fixed, the cardinality of the constant test set tends to be minimal or near minimal. A ripple-carry adder (RCA) composed of full-adder FUBs is probably the best-known example of a regular circuit. It is also C-testable with just eight tests, the minimum number possible. Figure 2 illustrates the eight tests for an RCA of arbitrary size. The binary values inside each full-adder FUB shows the values applied to the carry input, and operand inputs, respectively. Each test consists of a repeating pattern of eight full-adder input-pattern combinations, each corresponding to a row of the truth table of the full-adder function. Note that each test \( i+1 \) is a left-shifted version of test \( i \), a property that makes BIST extremely efficient.

For the CM-LCV, we chose a two-dimensional array structure since such a structure improves diagnosis due to increased diverging fanout for signals that are further away from the FUB-array outputs. Although the necessary and sufficient conditions for two-dimensional array C-testability is unknown, there is a plethora of sufficient conditions that ensure two-dimensional array C-testability. One simple approach that ensures C-testability relies on the use of a FUB function that is bijective. A bijective Boolean function is a lossless function where every input-pattern combination maps to a unique output-pattern combination. The first two columns of Table 3 shows an example 2-input, 2-output bijective function that has one vertical and one horizontal input, and one vertical and one horizontal output. It can be easily shown that a two-dimensional array that has an \( n \)-input bijective FUB function is C-testable for all IP faults with the minimum number of test patterns, namely \( 2^n \). Figure 3 shows one of the \( 2^n = 4 \) test patterns of a \( 5 \times 3 \) two-dimensional array of FUBs that implement the function of Table 3. Each FUB block shows a pair of input patterns (in decimal). The top value is the fault-free value established by the test pattern. The bottom value shows the erroneous pattern (if any) that is created assuming that the FUB located at position (2,3) suffers from the IP fault \( 2 \rightarrow 0/3 \) (described by the 3rd column of Table 3). The erroneous values produced are widely propagated to the outputs of all the shaded FUBs in Figure 3.

Table 3: A 2-input and 2-output bijective function.

<table>
<thead>
<tr>
<th>( i,d_h )</th>
<th>( o,v_{oh} ) (expected)</th>
<th>( o,v_{oh} ) (faulty)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00( \rightarrow 0 )</td>
<td>11( \rightarrow 3 )</td>
<td>11( \rightarrow 3 )</td>
</tr>
<tr>
<td>01( \rightarrow 1 )</td>
<td>10( \rightarrow 2 )</td>
<td>10( \rightarrow 2 )</td>
</tr>
<tr>
<td>10( \rightarrow 2 )</td>
<td>00( \rightarrow 0 )</td>
<td>11( \rightarrow 3 )</td>
</tr>
<tr>
<td>11( \rightarrow 3 )</td>
<td>01( \rightarrow 1 )</td>
<td>01( \rightarrow 1 )</td>
</tr>
</tbody>
</table>

Having established the appropriateness of bijectivity for the FUB, we now turn to the selection of which bijective function to implement given the general goal of optimizing both diagnostic resolution and accuracy. A bijective function of \( n \) bits is equivalent to a permutation on all of the possible output combinations. Thus, the total number of \( n \)-input bijective functions is \( (2^n)! \). Previous work [25] focuses on a subset of bijective functions based on the concept of sub-function injectiveness. Given a binary function \( F(i_v,i_h) = (o_v,o_h) \), its vertical and
horizontal output functions can be partitioned into two separate binary functions \( F_v(i_v, i_h) = \alpha_v \) and \( F_h(i_v, i_h) = \alpha_h \), respectively, that is, \( F(i_v, i_h) = (F_v(i_v, i_h), F_h(i_v, i_h)) \). The vertical function is vertically injective if each vertical input produces a unique output regardless of the horizontal input, that is, \( \forall i_v, \forall i_v \neq i'_v; F_v(i_v, i_h) \neq F_v(i'_v, i_h) \). Horizontal injectivity is defined analogously. A FUB function constructed from vertical and horizontal injective functions is guaranteed to propagate an error at its vertical (horizontal) input to its vertical (horizontal) output; thus, an array of FUBs implementing such a function guarantees error propagation along the rows and columns. In the worst case however, an error would only propagate to a single row or column output which would potentially limit resolution.

To improve diagnostic resolution, a greater constraint is placed on the bijective function. Specifically, both the horizontal and vertical component functions are required to be simultaneously vertically and horizontally injective. We call such a bijective function \( \text{vh-bijective} \) (vertically and horizontally bijective). A vh-bijective function propagates an error at either of its inputs to both its horizontal and vertical outputs, that is:

\[
\begin{align*}
\forall i_v, \forall i_v \neq i'_v; & F_v(i_v, i_h) \neq F_v(i'_v, i_h) \quad \text{and} \quad F_h(i_v, i_h) \neq F_h(i'_v, i_h) \\
\forall i_v, \forall i_h \neq i'_h; & F_v(i_v, i_h) \neq F_v(i'_v, i'_h) \quad \text{and} \quad F_h(i_v, i_h) \neq F_h(i'_v, i'_h).
\end{align*}
\]

Thus, in an array that has a vh-bijective FUB function, an error propagates both along the row and column, thus enabling near-ideal diagnostic resolution (as demonstrated in the next section). However, it is not readily apparent that vh-bijective functions actually exist. A constructive algorithm was used to create all possible 2-bit \( F_v(i_v, i_h) \) and \( F_h(i_v, i_h) \) that possessed the required properties (i.e., vertical and horizontal injectiveness). These vertical and horizontal functions were then exhaustively combined to form 4-bit functions and tested for bijectivity, resulting in the identification of 6,912 vh-bijective functions. All 6,912 functions were synthesized and ranked based on logical complexity, and two (“gamma” with low complexity, “beta” with moderate complexity) were arbitrarily selected for FUB implementation.

### 3.2 LCV Design

The logic regularity required for C-testability can be leveraged to dramatically improve both design time and design density, where density is simply defined here to be the number transistors per unit area.

At the logic level, simple scripts are written that take a netlist schematic of an \( n \)-input functional unit block (FUB) as input and the desired \( M \times N \) array size, and produces a behavioral-level and RTL description of the corresponding two-dimensional array of FUBs that form the LCV. The FUB itself can either be synthesized using commercially-available tools or manually mapped to any available standard-cells to reflect properties of customer designs. It should be noted here that multiple mappings of the FUB can be used to diversify the cells utilized in the LCV at the cost of a lower sampling rate, assuming a fixed number of LCV instances will be fabricated. In other words, the FUB array can be heterogeneous in nature in that several FUB logic designs can be used within the array. Figure 4 illustrates this situation where each different color FUB represents a different gate-level (and/or layout) implementation of its vh-bijective function. Due to the simplicity of this logic design flow, arrays for the CM-LCV consisting of hundreds of thousands or even millions of standard cells can be designed in virtually zero time.

The regularity of the logic level can also be exploited to create a very efficient layout. Essentially the layout of the FUB is created to mimic the regularity exhibited at the logic level. That is, horizontal and vertical input pins are positioned at the left and top column, thus enabling near-ideal diagnostic resolution (as demonstrated in the next section). However, it is not readily apparent that vh-bijective functions actually exist. A constructive algorithm was used to create all possible 2-bit \( F_v(i_v, i_h) \) and \( F_h(i_v, i_h) \) that possessed the required properties (i.e., vertical and horizontal injectiveness). These vertical and horizontal functions were then exhaustively combined to form 4-bit functions and tested for bijectivity, resulting in the identification of 6,912 vh-bijective functions. All 6,912 functions were synthesized and ranked based on logical complexity, and two (“gamma” with low complexity, “beta” with moderate complexity) were arbitrarily selected for FUB implementation.
FUB layout boundaries, respectively. Similarly, horizontal and vertical outputs are positioned at the right and bottom boundaries, respectively. Such an arrangement allows the layout of an $M \times N$ FUB array to be created through simple FUB abutment. As mentioned, the FUB itself can be either manually designed or place-and-routed using commercial flows. Either way, creating the final layout of the FUB array is straightforward through FUB abutment.

A FUB array design based on abutment may not accurately reflect the same back-end of the line (BEOL) layout properties of an actual customer design. There are several approaches that can be employed to remedy this situation however. One straightforward solution is to use the normal commercial design flow to place-and-route (P&R) an RTL description of the FUB array. This approach incurs the time cost of physical synthesis and relinquishes some area efficiency of abutment based design but maintains the test and diagnosis characteristics described in the next section. A second approach that preserves design density and avoids placement cost involves perturbing the physical placement of the FUBs and using the P&R to complete the routing. Figure 4 illustrates this idea. Specifically, it shows a perturbed version of the 5×3 array of Figure 3 where FUBs that are logically adjacent (e.g., (1,1) and (1,2)) are no longer physically adjacent. For such a placement, P&R would create upper metal layer routes that would be reflective of actual customer designs.

![Figure 4: Placement of FUBs to force P&R to create upper-level metal routes that mimic actual products. The different FUB colors represent different logic and/or physical implementations of the FUB function.](image)

### 4 Design and Test Experiments

Here, the design and test properties of several fully-implemented (logic and layout) CM-LCV designs are described. Specifically, in Section 4.1, various FUB and FUB-array implementations of the bijective function described in Section 3.1 are described and compared. Similarly, the testability and diagnosis properties of FUB arrays that use the constant test set of 16 tests are investigated, and compared to test sets generated by a commercially-available ATPG tool.

#### 4.1 LCV Test and Diagnosis Analysis

Here, the performance of traditional commercial ATPG is compared and contrasted with the constant-test methodology using various CM-LCV designs. In the following analyses, arrays constructed from a gate-level implementation of the “gamma” vh-bijective function (i.e., FUB,) are used. First, test set size and detection characteristics of the C-testability test set is compared with a standard single-detect stuck-at test set generated for these arrays using commercially-available ATPG. Table 4 compares the ATPG test-set size with the constant-test set as a function of the array dimensions. Note that the test set generated by our methodology is of constant size (16 vectors) irrespective of the array size and requires virtually no computational effort to generate. The reported ATPG test-set sizes are the minimum of three runs with different random seeds for each run. It should also be noted that elimination of the random test generation phase of ATPG significantly increased both the run time and test-set sizes. Although the ATPG test-set sizes are small, they are approximately 2X larger than the constant test set and require several orders more of CPU time to generate.

<table>
<thead>
<tr>
<th>Array size</th>
<th>No. of gates</th>
<th>No. of SSL faults</th>
<th>ATPG No. of tests</th>
<th>Time (sec)</th>
<th>Constant test No. of tests</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 × 10</td>
<td>1860</td>
<td>9680</td>
<td>20</td>
<td>0.01</td>
<td>16</td>
<td>0.12</td>
</tr>
<tr>
<td>25 × 25</td>
<td>11025</td>
<td>58700</td>
<td>29</td>
<td>0.16</td>
<td>16</td>
<td>0.13</td>
</tr>
<tr>
<td>50 × 50</td>
<td>43,3K</td>
<td>232,4K</td>
<td>32</td>
<td>2.4</td>
<td>16</td>
<td>0.12</td>
</tr>
<tr>
<td>100 × 100</td>
<td>171.6K</td>
<td>924.8K</td>
<td>37</td>
<td>53.7</td>
<td>16</td>
<td>0.15</td>
</tr>
<tr>
<td>250 × 250</td>
<td>1.1M</td>
<td>5,762M</td>
<td>42</td>
<td>2046</td>
<td>16</td>
<td>0.20</td>
</tr>
<tr>
<td>500 × 500</td>
<td>4.3M</td>
<td>23.0M</td>
<td>46</td>
<td>&gt;18000</td>
<td>16</td>
<td>0.22</td>
</tr>
</tbody>
</table>

Table 4: LCV design and test characteristics for FUB arrays of various sizes.

The stuck-at fault detection characteristics for the constant test set is compared to the ATPG test set for a 25×25 array. (Similar results are observed for the other array sizes.) All stuck-at faults are fault
simulated for each test in the two test sets. Detection counts (i.e., the level of N-detect [26]) for each fault are presented in the histogram of Figure 5. Although the ATPG test set is almost 2X the size of the constant test set, a better minimum detect threshold (all stuck-at faults are detected at least four times) and a much tighter distribution of N-detect is achieved using the constant test set. It is important to note that, while the N-detect distribution observed for the constant-test set is dependent upon the gate-level implementation of FUB function, the constant-test set minimally and exhaustively tests each FUB instance which means it covers the array uniformly and will never have an irredundant stuck-at fault that escapes detection.

The diagnostic resolution for the two test sets is also compared. Specifically, a random sample of 613 stuck-at fault simulation signatures are diagnosed using the constant and ATPG test sets, respectively. Figure 6 is a histogram of the diagnostic resolution (defined as the number of candidate faults returned by a commercially-available diagnosis tool) for this sample. The resolution distributions are virtually identical, with the only exception being the improvement from a resolution of two to one for a single stuck-at fault for the ATPG test set. In summary, the C-testable test set for the 25x25 FUB array resulted in a much smaller test set size (the ATPG test set is ~2X larger), a tighter N-detect distribution, and virtually identical diagnostic resolution with respect to tests generated using commercial ATPG over much longer run times.

The results for other types of fault models improve beyond what was observed for the stuck-at fault model. For example, the input fault (IP) model [19] is a functional fault model that is applicable to gates, complex cells and sub-circuit blocks such as the FUBs used in the CM-LCV. Used extensively in the regular circuit test literature, the IP fault model essentially assumes that the function of a combinational block under test can change to any other combinational function. A change in block function can be easily captured as one more IP faults, each of which is denoted as $ip \rightarrow o/o'$, where $ip$ is an input pattern to the block, $o$ is the expected response (which can be multiple bits), and $o'$ is any erroneous response where $o\neq o'$.

The IP fault detection characteristics for the ATPG and constant-test sets for a 10x10 array are investigated. Each FUB in the array is susceptible to 240 (16 input patterns x 15 possible faulty output responses) IP faults, which means there is a total of 24,000 possible IP faults in the 10x10 FUB array. All of these faults are simulated for each test in the two test sets. The amount of N-detect for each fault is
tabulated for the two test sets and is presented in the histogram of Figure 7. Again we observe that the smaller, constant-test set outperforms the ATPG test set. Specifically, the constant-test set exhibits perfect $N$-detect=1 behavior for all of the FUB-level IP faults in the array while the ATPG test set fails to detect a significant proportion of the IP faults (6,540 out of 24,000 are undetected, thus achieving a fault coverage of only 73.1%). Similarly, Figure 8 shows that the diagnostic resolution is superior when IP fault responses are diagnosed using the constant tests, which is attributed mainly to the superior fault coverage of the constant-test set.

Finally, to demonstrate the superior diagnosis characteristics of the vh-bijective function over a simple bijective function, we perform various diagnosis experiments. Specifically, two 10×10 arrays of FUBs, one using the "gamma" vh-bijective function and another using a randomly-selected non-vh-bijective function (henceforth referred to as "alpha") are analyzed. A random sample of 237 different IP fault responses are diagnosed using the corresponding constant-test sets for each array, respectively. Figure 9 is a histogram of the diagnostic resolution for each array. Generally, the array based on the gamma FUB function produces superior resolution due to the absence of the long resolution tail exhibited by the alpha FUB function.

4.2 LCV Logic and Layout Analysis
Two versions of a 10×10 CM-LCV are created, a manual design based on gamma function FUB$_g$, and a synthesized version that is also based on FUB$_g$. Both designs are created using a commercially-available 65nm standard-cell library with a single scan chain for controlling and observing the array inputs and outputs, respectively.

The manual design is built using double-height cells. The FUB consists of 18 NAND and 5 INV gates. The upper and lower halves of the cell positioning are somewhat symmetrical consisting of 9 NANDs and 2 to 3 INVs each. The 9 NAND gates are grouped into 6 groups of 2 NANDs feeding a third. The gamma FUB$_g$ could not be made to be vertically symmetrical due to an odd number of INVs.

As shown in Figure 10, horizontal inputs (A and B) enter the FUB from the left, and the two vertical inputs, C and D, enter the block from the top. The horizontal outputs W and X connect directly through abutment to the neighboring FUB’s inputs A and B, respectively. Similarly, vertical outputs Y and Z connect to the inputs C and D of the neighboring FUB below.
overlap is possible with the NANDs. The inputs and outputs are aligned so the FUBs “snap” together through abutment eliminating routing after placement as shown in Figure 10. The design uses the first three metal layers. Metal-2 is horizontal and is mostly unidirectional with two small wrong-way jogs. Metal-3, for the most part, runs vertically between the top and bottom halves of the FUB.

Figure 11: Manual FUB, array design.

Figure 12: Placed-and-routed FUB, array design.

A modular template is created so that a different FUB function and/or layout could be swapped in without any rework or rewiring. The largest FUB will dictate the grid size for the template. Different FUB designs would simply need to follow the template. Double-height cells eliminate the need to flip cells (MX) every other cell to match VDD and GND.

As mentioned in Section 3.2, product-like BEOL routings can be achieved by removing wires within the FUBs and replacing them with long, upper-layer routes. These routes can even expand past the FUB boundaries and return to complete the connection. A significant number of these traces can simulate realistic BEOL conditions.

A commercial flow is used to create a place-and-routed version of the CM-LCV. The synthesized FUB, implemented with 12 NANDs and 5 INVs that has a layout footprint of 7.0µm×5.2µm versus the 6.5µm×5.2µm footprint of the manual design. The commercial flow uses localized placement that led to a reduced routing length but also added a total of 322 filler cells. Some of these filler cells could be due to using a core utilization of 90%. The FUB, array based on the manual design is almost 25% smaller however at 66µm×53µm as compared to the 68.3µm×68µm due to the efficiency of the abutment-based design.

5 Conclusions and Future Work

The regular architecture of the CM-LCV (Carnegie-Mellon Logic Characterization Vehicle) has a number of distinct advantages. First, synthesis of the gate-level netlist is trivial in that a simple script can be used to easily generate a design instance of any desired size and layout footprint. The same is true for layout synthesis. That is, once the basic functional unit block (FUB) is constructed for abutment-based connections, it can be easily snapped to other FUBs, or to the compatible scan-chain blocks. Second, the challenge of ATPG is virtually eliminated since, by construction, the n-input vh-bijective function implemented in the FUB ensures that any N-by-M array constructed from NxM instances of the FUB is completely testable for all input pattern faults [19] with the minimum number of tests (2^n), a property known as C-testability [5]. In addition, experiments with commercial ATPG tools reveals that number of test patterns grows slowly with the size of the array. This characteristic is advantageous in case the normal tool flow is preferable but results presented here show that the constant-test sets have superior test and diagnosis characteristics. Finally, given the inherent fanout that exists within the array, logic diagnosability (resolution, accuracy, etc.) is extremely high both with an approach that exploits the inherent regularity and the conventional approaches found within existing commercial tools.
More, due to the regularity, physical diagnostic resolution is also dramatically improved. All of these advantages and more are easily obtained since the functionality of FUB is now unrestricted and thus can be geared towards maximizing information extraction while still exhibiting crucial characteristics of product-like (i.e., customer) designs. Current work is focused on adding a comprehensive BIST to the CM-LCV. We are also currently working with industrial collaborators to fabricate a CM-LCV with a moderate volume in a state-of-the-art technology.

References


