Built in Test Coverage and Diagnostics
Best Practices to Achieve Built in Test Success

Jeffrey Smith
Agilent Technologies
Product Marketing Engineer
Loveland, CO 80537
jeffrey_d_smith@agilent.com

Duane Lowenstein
Agilent Technologies
Test Process Analyst
Andover, MA 01810
duane_lowenstein@agilent.com

Abstract – Ideally, everyone would like to embed test into their products today, whether it’s to reduce costs, build in support capabilities, ensure better quality, reduce dependency on expert technicians, or for one of many other intelligent motivations. However, it seems the technology of Built in Test (BIT) has not taken off or been adopted like many believed it would. The reasons for this poor adoption are diverse: some blame it on design, others fault product specifications and even others blame it on the fundamentals of the technology itself.

The underlying root cause of slow BIT adoption may be a result of an inaccurate belief that designing an embedded test is more difficult than designing any other test. In reality, understanding what to embed is simply a function of knowing the coverage you are seeking and at what level you want to be able to diagnose the problem when a failure occurs.

Presented in this paper are several considerations on how to best approach Built in Test. This preferred approach is illustrated by explaining the importance of understanding how to achieve effective and appropriate coverage levels, while also maintaining a useful level of diagnostic resolution. Understanding and application of these concepts will effectively enable optimal design and implementation of Built in Test.

I. INTRODUCTION

It wasn't long ago that the idea of incorporating Built in Test into a product seemed unnecessary, costly, and likely to over-complicate a relatively easy to diagnose system. As time has marched forward, the ever increasing complexity of circuits and devices has begun to demand a more involved and intimate test relationship with how we approach our devices.

This increased device intricacy not only complicates our testing methodology, it also demands a new level of understanding of said methodology. Developing this understanding of our testing strategy does require more costs in time and effort upfront. However, this really is a situation where a minute of invested time at the front-end will result in an hour gained on the back-end. [1]

Developing a testing strategy and system around BIT may seem counterintuitive to some designers as it adds increased complexity to circuits and will usually results in increased device fixed costs. However, with a proper Built in Test solution, variable costs will undeniably drop as support and production costs decline as a result of production volume and overall life cycle management savings. These drops in variable cost nearly always outweigh the increased fixed cost incurred when implementing a BIT solution.

To realize these net savings the Built in Test solution needs to provide adequate coverage for device components while also being able to identify specific components at fault. With these practices in place, BIT effectively changes the embedded situation from “not knowing what is being tested” to “knowing exactly what is being tested and what component is at fault when a failure is reported.” The ability to quickly and easily identify a specific component at fault is the fundamental promise of an effective BIT system. This capability sets up all other successes that can be achieved with BIT.

II. TEST COVERAGE

A. What Coverage Is and Is Not

Coverage can be defined as the degree to which a device under test is exercised by a test suite [2]. Said another way, coverage is the extent that a device is actually being evaluated for problems by a given test strategy. Generally speaking, the higher the coverage obtained the better the overall test suite is at determining if there is a fault in the system.

Coverage alone does not provide the capability to identify the cause of a fault; rather it is only the ability to know that a fault exists. In other words, high coverage amount does not guarantee any abilities of a test strategy to isolate particular faults nor does high coverage necessarily represent a more ideal test suite. While “higher” coverage does increase the ability to find the existence of a fault in a device, the tests necessary to achieve said high coverage amounts may be difficult to conduct, costly, or even impossible. Therefore no hard coverage level can be specified in advance. The right amount of coverage is relative to a given device and must be evaluated independently through a series of classical design trade-offs.

A simple example of this trade-off concept in action would be to consider the case of testing an RF board without...
understanding coverage amounts. A test can be conducted that measures the output of the RF board that “validates” the board is working. However, this test value is limited in its worth because it is does not describe to what level the board components were being utilized (exercised) to produce the “valid” test data. Under an alternate situation where the board was being exercised differently a failure result may occur. Having a testing strategy that is capable of quantifying coverage amount enables the designer to determine the likelihood of this scenario occurring, and whether adding additional tests is worthwhile to reduce the risk of failing to identify component failure.

B. Why Coverage is Important to Built in Test

The degree of coverage exercised is critical to quantify and understand in order to be successful with Built in Test. Without a clear comprehension of coverage amount in a Built in Test solution the BIT system is effectively “flying blind” with regards to the system’s intent. This failure of a BIT system is not inherent to BIT and is instead a direct result of the test designer’s failure to account for proper coverage in the test strategy.

Due to the common practice of test engineers designing tests from a specific test perspective instead of a component perspective [3], BIT solutions with unknown coverage can and likely will result in a solution that fails to adequately examine the applicable areas of concern on a device under test (DUT). This makes the extra effort to put BIT in place somewhat useless and obviously undesirable. Fortunately, when a test engineer begins to fully account and quantify the coverage amounts there is a natural conversion from the test specific perspective to the component perspective. This conversion in thought process is the basis of a coverage intelligent approach which is necessary to be successful with BIT, paving the way to take full advantage of the excellent cost savings BIT provides.

It is through this strong understanding and optimization of device coverage that the benefits of a BIT system are maximized directly resulting in an improved bottom line for the project. Coverage optimization also ensures that a project is more easily managed through the project’s life cycle; clear coverage understanding enables key stakeholders to effectively set metrics and goals for the coverage amounts necessary to guarantee the desired level of quality while also providing a workable path for the designer to achieve success. In order to effectively design tests and achieve these results however, it’s necessary to also understand what level of diagnostic resolution you have available; it will directly affect the type and amount of tests utilized to achieve optimal coverage.

III. DIAGNOSTIC RESOLUTION AND CAPABILITY

A. Desired Diagnostic Resolution

A truly successful Built in Test solution will not only include the capability to recognize that there is a fault (coverage), but also the ability to isolate and identify what the cause of the problem is (diagnostics). While simply identifying that a fault exists in a device may be sufficient in some cases, a failure to build in the diagnostic capability necessary to identify the root cause may limit quality efforts by preventing the fault cause from being fixed in existing and future devices.

It is commonly accepted that isolation of faults is better done than not, but what is often poorly understood is to what level diagnostics need to be developed to be effective. Again, we have an area that cannot be absolutely defined as it will depend on the device itself, its operating environment and the intent behind the BIT system. What we can do however, is identify some useful guidelines for what level the diagnostics should be developed to in BIT solution.

Specifically, the level of diagnostics that should be sought after in most BIT systems is the level that conveys the appropriate actionable information. In other words, the diagnostic capabilities of the BIT system should communicate the information necessary to properly address the fault for the given situation. It may be useful then to work backwards from the desired level of information to the diagnostic test.

For many devices with a BIT system, this means that the diagnostic resolution should be at the serviceable board level. For example, there is no need to have a diagnostic resolution level reach to individual capacitors if the capacitors themselves are not serviceable in a given device. As another example, take the case where a device is 100% unserviceable, such as some sort of deployed unit. The resolution level only needs to be at the level where the information that is returned from the BIT system indicates whether the device is still functional or should be abandoned.

B. Predictive vs Historical Diagnostic capability

In constructing the diagnostic capability of a Built in Test system there are two different methodologies that may be pursued: the predictive diagnostic modeling method and the historical diagnostic modeling method [4].

The predictive modeling method benefits from being able to be applied to new designs before a device is ever constructed or a design is finished. Such modeling enables the device’s design to be positively affected while still in the design stages, something that is not possible with the historical modeling approach. Furthermore, good predictive modeling benefits from being able to render diagnosis information without the need of experiencing costly real failures. Predictive modeling works just as the name implies, by being able to predict where problems are most likely to occur.

Historical modeling on the other hand can be optimally employed in designing BIT systems where a new device design is based largely on an existing design. In the case of building an evolutionary design ample amounts of historical data can be applied, thus saving time and money in developing the model. Historical modeling makes more sense in this case even though it lacks some of the predictive modeling advantages. However, care must be taken when implementing a historical modeling approach to make sure the data utilized is truly of high enough quality to be applied.

Understanding if a device that is to receive a BIT solution is a wholly new device, or based on an existing device with historical data available, will assist a designer in choosing what diagnostic methodology to pursue. Some designers will be
inclined to pursue a common approach regardless if it is a new device or an evolutionary one as they will prefer to stick with what they are most familiar with. This inclination to stick with a common approach, regardless of situation, should be managed to ensure that the best approach is utilized in each case. Because incorporating a BIT system usually requires significant new design work, a predictive diagnostic modeling approach is most frequently the appropriate approach to apply when incorporating BIT.

IV. BALANCE BETWEEN COVERAGE AND DIAGNOSTICS

A. A necessary and powerful relationship

As stated earlier, coverage enables your BIT system to identify if a problem exists (detection) where diagnostics enables the system to identify the source of the problem (isolation). Each capability is unique and shares important information, but they are also very reliant on one another. The dependency is a feedback loop type relationship where coverage relies on diagnostics to be able to determine the cause, while diagnostics relies on coverage to determine that there is a fault. That is, you need both working in concert together to fully realize the benefits that a BIT solution provides.

To visualize this relationship, imagine a person’s visit to a doctor for a physical exam. The tests the doctor conducts to evaluate the person’s health are the equivalent of test coverage. If some tests come back negative the doctor would attempt to diagnose (i.e. isolate) the cause of the negative tests based on the test data. Should the doctor not have enough test information (coverage) to diagnose you the doctor may expand the test coverage to be able to reach a diagnosis.

In another illustration, consider that the visit to the doctor is just a routine physical; the tests conducted are likely to be relatively simple and cost efficient. This scenario could be described as having low coverage and thus it’s entirely possible that the doctor may miss the fact that the patient has a brain tumor. The point is that sufficient coverage is necessary for accurate diagnostics and without adequate diagnostic capabilities coverage is limited in its usefulness. However the coverage necessary to find out if you a brain tumor may be too costly or time consuming to conduct on every physical.

This dependent relationship between coverage and diagnostics is critical to achieving success with BIT and cannot be underestimated in its importance. The relationship is so significant that the tools we utilize to incorporate BIT must recognize the association and provide support for managing the relationship. A failure to utilize intelligent tools makes the process of incorporating BIT into a device significantly more difficult, setting up designers for failure and preventing BIT from realizing its true potential.

V. ACCESSIBLE SOLUTION

A. Agilent Fault Detective

One solution that is built on the principals presented in this paper is Agilent Technologies’ Fault Detective (FD) software suite. It is based on a predictive modeling environment utilizing Monte Carlo simulation that is purpose built to manage the reciprocal relationship between coverage and diagnostics. Because the application fully acknowledges the dynamics shared between coverage and diagnostics the software is a powerful solution for BIT designers looking to manage their process.

Using Fault Detective’s intuitive interface, users are able to quickly develop device models and input test strategies, thereby enabling users to understand their coverage, their diagnostic resolution and most importantly, how each affects one another. The speed at which designers are able to work within the application is critical to the users’ goals; being able to witness the effects of test additions in real time results in superior testing methodology. Fault Detective fully supports an interactive environment where cause and effect can quickly be identified and exploited.

B. Achieving Coverage through test optimization

In working to enable users to understand the true extent of their coverage, Fault Detective identifies the contribution of individual tests in the suite and assigns an improvement score for all possible faults. By analyzing individual tests, FD allows the user to understand which tests may fail to contribute to the overall ability to detect and isolate faults, as well as to identify those tests that are redundant. These “suspect tests” are quickly indentified and drawn to the attention of the user to aid in optimization. Fault Detective then takes this capability one step further by assigning improvement scores to each possible fault, shown in Figure 1 below.

<table>
<thead>
<tr>
<th>Fault</th>
<th>Detection Max Improvement</th>
<th>Isolation Max Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>20.2%</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>EPROM_lower</td>
<td>9.6%</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>EPROM_upper</td>
<td>10.4%</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Flash_lower</td>
<td>1.1%</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Flash_upper</td>
<td>1.2%</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>DAC</td>
<td>10.0%</td>
<td>0%</td>
</tr>
<tr>
<td>12VDC</td>
<td>50.1%</td>
<td>&lt;1%</td>
</tr>
</tbody>
</table>

Figure 1. Test Suggestion Analysis

The improvement scores provide the user an idea of which faults are not easily identified with the current test strategy. Improvements can then be made to the test suite in order to improve the detection and isolation scores as appropriate for achieving the desired level of coverage. In this example, the Detection Max Improvement score of 50.1% for component 12VDC indicates that significantly more coverage is possible if the designer felt it was necessary. Conversely, the Isolation Max Improvement scores are indicating little improvement is possible for any of the components in this simplistic model. Through this optimization, the user can have confidence that their strategy’s coverage maximizes the ROI of their test suite.

C. Enabling effective diagnostic resolution

Fault Detective’s predictive model-based diagnostics use a high-level model to determine the likely cause(s) for failed
functional tests. The underlying method for determining faults is based on Bayesian logic and is uniquely powerful and accurate in its capabilities. The software, using the functional relationship between components defined in the model and test result data, determines fault causes with a speed and accuracy beyond that of any expert technician.

FD requires only a functional knowledge of the device under test thus enabling models to be developed from something as simple as a block-level perspective. The program provides users the capacity to define models at different functional levels (device, board, component, etc.) thus allowing the application to easily scale to the desired diagnostic resolution level. Figure 2 below illustrates a simple digital model with a resolution level to the subcomponent level. In this model, FD is capable of rendering a diagnosis to the specific subcomponent when a fault occurs, assuming enough test coverage is provided.

Because FD utilizes the functional relationship between components in the device, it has no requirement for detailed hardware specifications. The result of leaving out unnecessary information is a high-level model developed from a basic understanding of the device’s structure which can be assembled quickly and easily, paving the path for faster success.

D. Balancing Coverage & Diagnostics with Fault Detective

Fault Detective enables engineers to understand exactly the level of coverage they have and to what level they can identify specific causes of failure. Through this quantifying of the data and test suite, designers begin to explicitly see the relationship that coverage and diagnostics have on one another. Once this bridge of understanding is built, designers are empowered to make the necessary decisions in determining what kind and the number of tests that are necessary for the coverage and diagnostic isolation desired. Figure 3 illustrates a summary view within FD that shows overall predicted performance of both coverage (detection) and diagnostics (isolation) of a given test suite when paired with its device model.

<table>
<thead>
<tr>
<th>Predicted Performance</th>
<th>Detection</th>
<th>Isolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>85.0 %</td>
<td>85.0 %</td>
</tr>
<tr>
<td>Delta</td>
<td>3.4 %</td>
<td>14.6 %</td>
</tr>
</tbody>
</table>

This summary perspective provides the designer with a quick way to identify if the test suite is sufficient for the purposes of BIT and if it meets internal coverage and diagnostics standards. As a general rule, detection and isolation scores of above 85% are usually sufficient for most BIT environments [1]. Pursuing perfect scores often results in entering a diminishing return situation.

VI. SUMMARY

In the quest to achieve effective Built in Test for devices today’s designers must work to fully understand the intricate relationship that diagnostics and coverage have with one another. Through careful management of this relationship and practical application of commercially available tools like Fault Detective, there is little reason that Built in Test cannot be successfully incorporated into many more devices than we see today.

GLOSSARY

BIT  Built In Test
BIST Built In Self Test
DUT Device Under Test
FD Fault Detective
ROI Return on Investment

ACKNOWLEDGMENT

1) Thanks to Laura Johnson, Ken Colasuonno and Alan Copeland of Agilent Technologies

REFERENCES