Is Intelligent Testbench Automation For You?

by Mark Olen, Product Marketing Manager, Mentor Graphics

INTRODUCTION

Intelligent Testbench Automation (iTBA) is being successfully adopted by more verification teams every day. There have been multiple technical papers demonstrating successful verification applications and panel sessions comparing the merits to both Constrained Random Testing (CRT) and Directed Testing (DT) methods. Technical conferences including DAC, DVCon, and others have joined those interested in better understanding this new technology. An entire course curriculum is available at the Verification Academy. And many articles have been published by various technical journals, including in this and previous Verification Horizons editions. So with all of the activity, how do verification teams separate out the signal from the noise? How do they know whether Intelligent Testbench Automation is applicable to their designs? How do they know whether they will experience the advertised gains in verification productivity, as shown in Figure 1 to the right? This article does not discuss how Intelligent Testbench Automation works, as indeed there are several sources for that information just mentioned. This article discusses where iTBA is best applied and where it will produce optimal results, as well as where it is not and will not. As with most emerging technologies, iTBA yields better results in some applications than others.

VERIFICATION SPACE

The more choices to be made during verification the higher the impact realized from Intelligent Testbench Automation. A design that has a large number of configurations, modes, operations, commands, functions, variables, payloads, and other parameters, is the best application for iTBA. To effectively verify this type of design verification teams must test all important combinations of these choices. Examples would include SoC designs with multiple processors, busses, and peripheral interfaces; complex bus fabrics with multiple masters and slaves, and arbitration schemes; routers and more. iTBA rarely returns less than a 10X gain in productivity when presented with a large verification space with a large number of choices. Its natural ability to remove unwanted redundancy while preserving desired randomness maintains the Constrained Random Testing

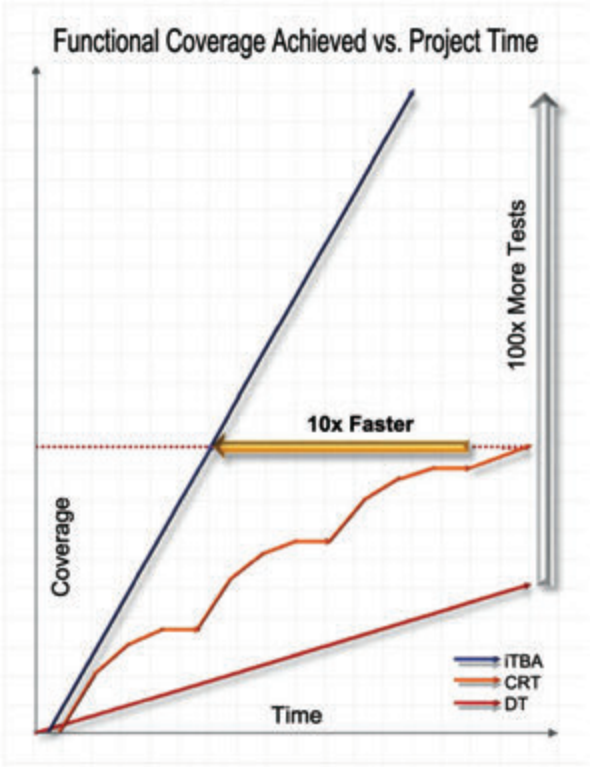


Figure 1 - Comparison of Functional Coverage Achieved Over Project Time

characteristic of generating tests for conditions not previously envisioned by verification engineers, while reducing the number of tests required to achieve or exceed coverage goals. In a perfectly balanced and symmetrical situation, iTBA achieves coverage goals faster than CRT according to the following equation -

Tests required by CRT to achieve coverage of N scenarios = $N \ln N + \gamma$

Test required by iTBA to achieve coverage of N scenarios = N

Where N equals the number of different combinations of conditions

And γ equals the Euler-Mascheroni1 constant = 0.5772[1]…

According to the equation above, once the verification space exceeds 22,027 scenarios, iTBA achieves a minimum

of 10X gain in productivity. As an illustration, please refer to Chart 1 below that shows a comparison between iTBA and CRT selection of a variable with 64 values. Chart 1 shows the selection distribution after 64 “tests”. The Graph-based iTBA selection achieves all 64 values in random order, while the Random CRT selection only reaches 42 values, missing 34% of the important values. Notice the uneven distribution of random selections in red, where 14 values have been tested multiple times, while 22 values have yet to be tested. If the variable values were re-ordered in the chart, we would in fact see a red bell curve distribution across the range, compared to the flat blue distribution generated by iTBA.

However in general, it would certainly be pretty aggressive to expect to achieve 100% coverage of all important scenarios with perfect efficiency, so please refer next to Chart 2 below that shows what happens when the simulation is allowed to continue for another 64 “tests”. Notice that the Random CRT distribution continues to repeat as expected, and that even after running twice the number of tests needed, CRT still reaches only 83% coverage with 12 corner cases still not verified. Meanwhile, if iTBA is allowed to continue running, all values are now tested twice. If you’re curious about how many tests it will take for Constrained Random Testing to finally reach those difficult to find corner cases, go back to the formula above and plug in N = 64. This experiment allowed the simulation to continue for 256 tests, at which point CRT achieved 96% coverage. So a single run of this simulation produced a result consistent with the formula above. And if the test were repeated several times with different seeds, the average result would converge to the formula’s calculation.

UNBALANCED CONDITIONS

While the testcases above are small in size, and most teams would not adopt a new technology for a 4x benefit, please keep in mind that this example depicts a perfectly balanced set of conditions. Each of the 64 values of our variable is equally likely to occur. For example, the protocol rule in Figure 2 generates transactions of varying burst lengths and payload sizes. The resulting protocol graph shown in Figure 3 shows a perfectly balanced protocol. If burst length is one, then there are three sizes of payload, for three equally weighted sub-combinations. And if burst length is two, three, or four, then there is only one valid size of payload, for three equally weighted sub-combinations, yielding six equally weighted total combinations.

Chart 1 - Coverage Results After Simulating 64 Tests

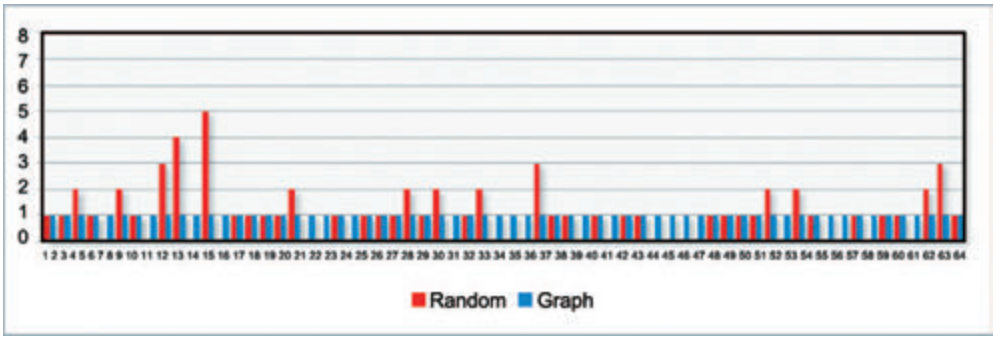
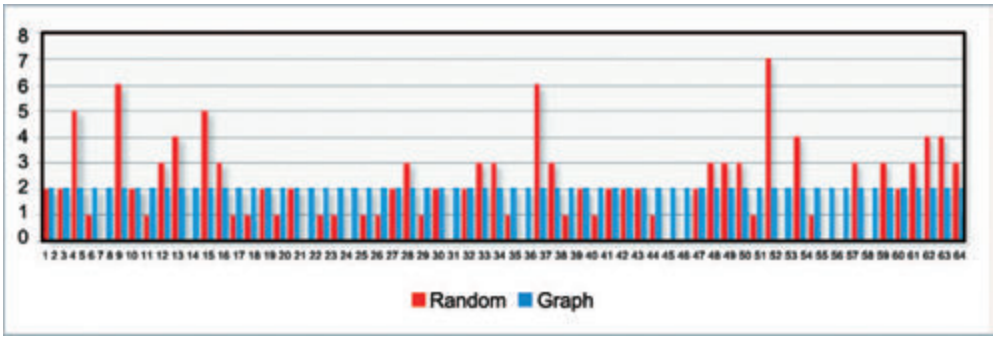


Chart 2 - Coverage Results After Simulating 128 Tests



```
simple_protocol = repeat {
    pre_fill
    addr
    rnw
    burst_len
    if {burst_len == 1} (size [1,2,4]) | if {burst_len > 1} (size [4])
    size
    post_fill
};
```

Figure 2 - Example Protocol Rule Description

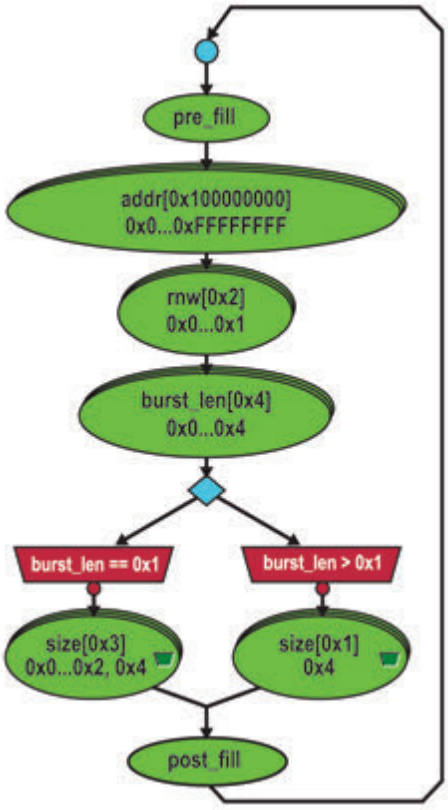


Figure 3 - Example Protocol Graph Description

Such a balanced protocol is very unlikely to exist (yet this one still yields a productivity gain of N ln N + γ). However, most actual verification requirements include difficult-to-find corner-case scenarios caused by unbalanced conditions. These can be design-related or test-related. Some design functions require very specific pre-conditioning sequences to enable rarely encountered functions (not depicted in the protocol above). In addition, some test goals require cover points that vary in size by orders of magnitude. If verification engineers are spending lots of time writing directed tests to fill coverage holes not tested by Constrained Random Tests, then these conditions likely exist. Intelligent Testbench Automation can return upwards of 100X gain in productivity when presented with very unbalanced conditions.

TARGET COVERAGE

There are many different types of coverage to measure during verification, but they often can be grouped into two main categories - black-box coverage and white-box coverage. Questa is better targeted at black-box coverages, which includes design stimulus coverage and design response coverage. These coverages can be derived from functional specifications, test plans, or engineering know-how. They are all related to how a design functions, and can be observed at the inputs and outputs of a design. White-box coverages include RTL code coverage, internal state coverage, assertion coverage, and clock domain coverage. These are best addressed by other key technologies included in the Questa functional verification platform. RTL code coverage is monitored and reported by Questa Simulation, while Questa inFact generates stimulus. Internal state coverage and assertion coverage are achieved by Questa Formal. And clock domain coverage is modeled, measured, and reported by Questa CDC. And all of these coverages, both black-box and white-box are collected by Questa Verification Management for rapid merging and reporting. In the end it is the integration of several key technologies that enables the Questa Platform to completely verify complex designs.

EXISTING ENVIRONMENT

The type of verification methodology already in place should also be considered when adopting Intelligent Testbench Automation. In each case iTBA can provide a step-function gain in productivity, but the steps taken to maximize the gain will be different. Consider the following three cases -

Case 1 - Constrained random test stimulus and functional coverage measurement are both in place, and simulation results and coverage reports are available. In this case the existing constraints and covergroups can be imported into the Questa inFact Intelligent Testbench Automation toolset. If the constrained random stimulus was achieving satisfactory coverage, then Questa inFact will be able to achieve the same coverage at least 10x faster as described

above, enabling much shorter turnaround time, or enabling test expansion to cover functionality previously thought not testable. If the constrained random stimulus was not achieving satisfactory coverage, then Questa inFact will offer even more value by achieving the target coverage while still reducing the time by 10x and eliminating the need to write directed tests to target the random resistant corner-case scenarios.

Case 2 - Constrained random test stimulus is in place, but functional coverage measurement is not. In this case the constraints can be imported into Questa inFact, and inFact's coverage editor can be used to target stimulus generation and report coverage achieved. In addition, Questa inFact can generate SystemVerilog covergroups to be used to report and manage coverage results with Questa Verification Management. The SystemVerilog covergroups generated by Questa inFact can even be used by Questa Verification Management to measure the effectiveness of the original constrained random stimulus, validating the step-function gain realized when using Questa inFact to generate stimulus.

Case 3 - Directed test stimulus is in place to select different test combinations and manage test sequencing. Typically there will be limited (or no) functional coverage measurement in place here. In this case the directed tests can be mapped into Questa inFact. Then inFact can automatically expand the number of test combinations without repetition. Questa inFact can also improve the test order, prioritizing important test combinations earlier in the simulation. In addition, Questa inFact can generate SystemVerilog covergroups to be used to report and manage coverage results with Questa Verification Management. Questa inFact will yield the best results when existing directed tests are short in duration, involving multiple variables with complex relationships. In these cases Questa inFact can quickly and efficiently multiply the directed tests by 100x or more, without repetition.

VERTICAL MARKETS

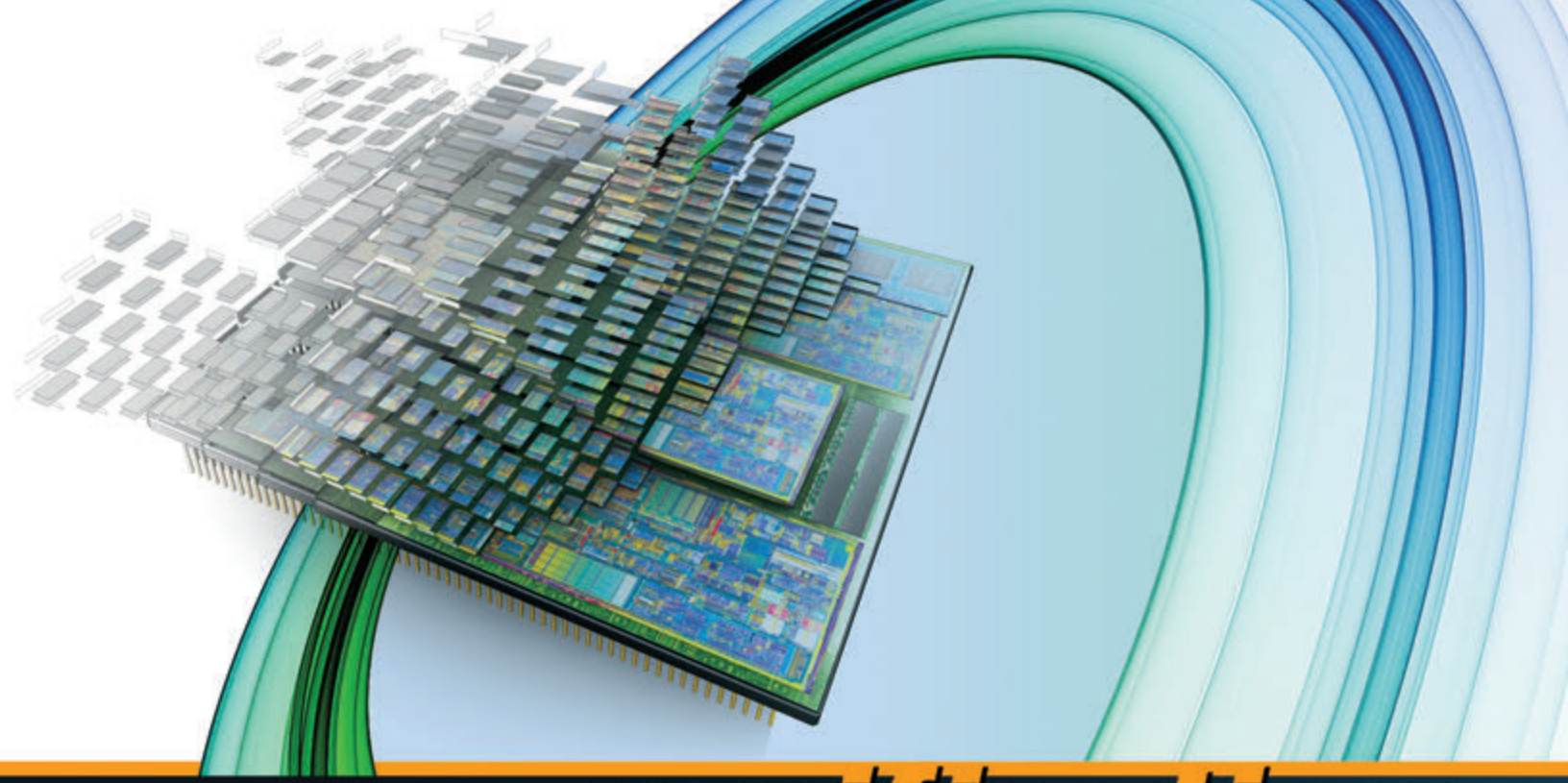
Design application segments have little direct impact on Intelligent Testbench Automation applicability. Telecom is not necessarily more applicable than consumer electronics, which is not necessarily more applicable than wireless, or mil-aero, or others. iTBA can be deployed at the block, subsystem, or system level to generate efficient stimulus and response that accelerates coverage closure, in spite of extreme design complexity. If there is any vertical market dependency, it would not be related directly to iTBA, but to other aspects of the verification environment. For example, iTBA is best employed with verification IP components to provide DUT-level interfacing such as bus masters and slaves, peripheral models, or transactors. Access to a ready supply of high quality verification IP will have more influence over vertical market applicability. That is one reason why Questa inFact is often deployed with Questa Verification IP, which provides high quality DUT-level interfacing, with multi-level viewing for ease of design debugging.

SUMMARY

Questa inFact's Intelligent Testbench Automation is most appreciated by customers who use constrained random testing, but then have to write directed tests to cover those hard-to-find corner-case scenarios. That's where Questa inFact shines. First, it helps verification engineers eliminate their wasted tests. Second, it helps them get to their coverage goals faster. Third, it relieves them from having to write directed tests to manually cover corner-case scenarios that are resistant to automated random testing. And fourth, it leaves them time to consider test expansion, thus enabling verification engineers to target even more functionality.

END NOTES

[1] The Euler Mascheroni constant (also called Euler's constant) is a mathematical constant recurring in analysis and number theory, usually denoted by the lowercase Greek letter γ (gamma). It is defined as the limiting difference between the harmonic series and the natural logarithm.



verification HORIZONS

Editor: Tom Fitzpatrick
Program Manager: Rebecca Granquist

Wilsonville Worldwide Headquarters
8005 SW Boeckman Rd.
Wilsonville, OR 97070-7777
Phone: 503-685-7000

To subscribe visit:
www.mentor.com/horizons

To view our blog visit:
VERIFICATIONHORIZONSBLOG.COM

Mentor Graphics®

www.mentor.com