

A Unified Verification Flow Using Assertion Synthesis Technology

by Yuan Lu, Nextop Software Inc., and Ping Yeung, Mentor Graphics Corporation

INTRODUCTION

As SOC integration complexity grows tremendously in the last decade, traditional blackbox checker based verification methodology fails to keep up to provide enough observability needed. Assertion-based verification (ABV) [1] methodology is widely recognized as a solution to this problem. ABV is a methodology in which designers use assertions to capture specific internal design intent or interface specification and, either through simulation, formal verification, or emulation of these assertions, verify that the design correctly implements that intent. Assertions actively monitor a design (or testbench) to ensure correct functional behavior. They detect design errors at their source, greatly increasing observability and decreasing debugging time.

Bugscope Assertion Synthesis

The ABV methodology is easy to adopt in most existing verification flows since it can be adopted incrementally. Besides capturing design intent manually with an assertion language such as system verilog assertion (SVA), an Assertion Synthesis tool, such as NextOp's Bugscope, can be used to create high quality assertions based on simulation activities [2].

Given a set of regression tests and the corresponding RTL, Bugscope generates properties which satisfy three requirements:

- true at every cycle of the simulation
- not easily implied by RTL only
- orthogonal to each other

These properties are further classified by designers as either assertions or cover properties. Assertions are checked in to increase verification observability while cover properties directly point to the missing functional coverage by simulation (see Figure 1). Intuitively, Bugscope captures the design/verification snapshot in the format of properties. Such information guides further verification. Therefore, Assertion Synthesis enables a progressive, targeted verification process, allowing design and verification teams to more easily uncover corner case bugs, expose functional coverage holes, and increase verification observability. The two advantages of Assertion Synthesis are

- Reduce manual assertion writing effort for designers which is believed to be the main hurdle for design team to adopt ABV;
- Synthesized cover properties provide a unique functional coverage report to the user;

In this article, we describe a unified verification flow by incorporating Bugscope into Mentor Graphics Questa/ Veloce verification flow. We will demonstrate how to integrate Bugscope with Questa simulation, Questa formal verification and Veloce simulation acceleration environment to achieve better observability.

Questa Simulation

Questa Simulation supports multiple verification methodologies including Assertion Based Verification (ABV), the Open Verification Methodology (OVM) and the Universal Verification Methodology (UVM) to increase testbench productivity, automation and reusability. It enables the automatic creation of complex, input-stimulus using scenarios described in terms of constraints and randomization using SystemVerilog or SystemC Verification (SCV) library constructs. Questa Simulation combines all of these forms of stimulus generation with functional coverage to identify the functionality exercised by the automatically generated stimulus. Using assertions as feedback for test



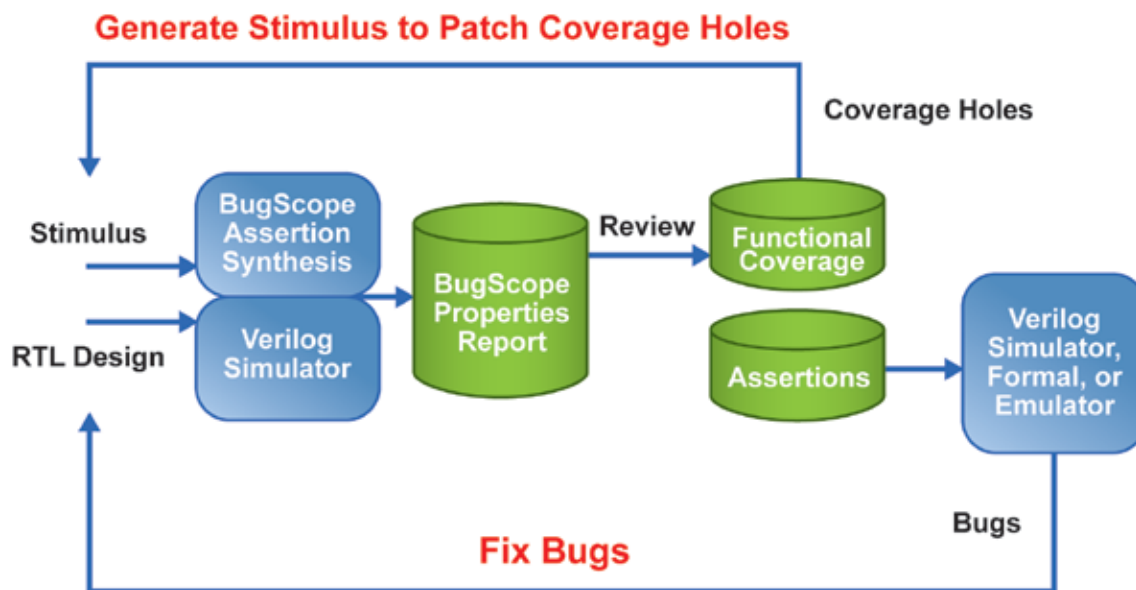


Figure 1 Assertion Synthesis Technology

creation, engineers can adjust constraints to focus random testing on coverage holes.

Questa Formal Verification

Questa Formal Verification supports general assertion-based formal verification to ensure that the design meets its specific functional requirements. With support for PSL, SVA, and OVL, including multi-clocked assertions, Questa Formal Verification easily verifies very large designs with many assertions. Its multiple high-capacity formal engines cooperate to complete verification faster. Questa Formal Verification is integrated with the Questa Simulation for easy debug of assertion failures.

Veloce Simulation Acceleration

Veloce Simulation Acceleration speeds up block-level and full SoC regression test runs by 100s to 1000s of times. It includes a simulation-like debugging environment, has 100% internal DUT visibility, and supports traditional break pointing and ABV. In a transaction-based acceleration environment, Veloce uses TestBench XPress (TBX) Software [3], and host-based transaction-level test benches to drive transactors in the Veloce system to drive the DUT. For test benches written in C/C++ or System C, TBX interfaces directly with the Veloce and executes the test

bench program. For SystemVerilog testbenches, TBX runs Questa on the host PC to drive the test bench through the Veloce-based transactors and DUT.

Questa Verification Management

Questa collects all coverage data — code coverage, assertions, formal, and functional coverage — into a single highly efficient Unified Coverage DataBase (UCDB) and makes them available in real-time within the testbench or for post-processing with Questa Verification Management. It can also capture information about the broader verification context and process, including which verification tools were used and even which parameters constrained these tools. The result is a rich verification history, one that tracks user information about individual test runs and also shows how tests contribute to the overall coverage objects.

A UNIFIED VERIFICATION FLOW WITH ASSERTION SYNTHESIS TECHNOLOGY

Given a testbench, no matter whether it is at block level or chip level, Bugscope generates assertions and cover properties based on the given tests. Expressed in SVA with binding statements ready, these property files are directly given to Questa simulator, Questa formal verifier or Veloce hardware accelerator to consume to reach intended coverage goals, check design intent, therefore reach verification signoff criteria (see Figure 2 on the following page).

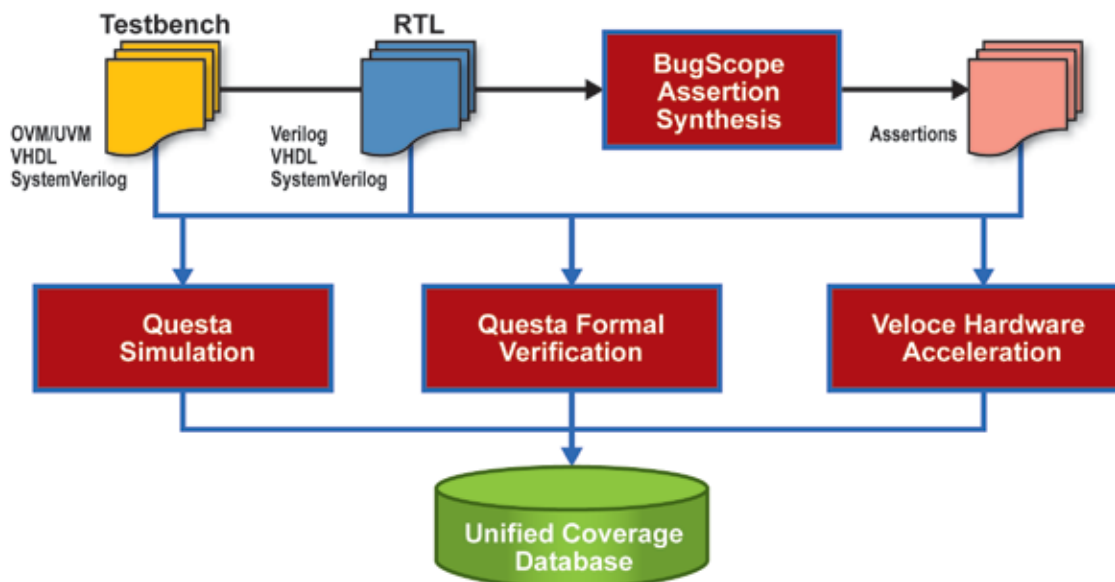


Figure 2 Unified Verification Flow with Assertion Synthesis Technology

Unified Simulation Flow

Given a block level testbench, Bugscope generates assertions and cover properties in SVA format. The cover properties point to functional corner case holes which are missed by simulation. These SVA properties can be directly included in Unified Coverage DB which is consumed by Questa simulator to guide the user to reach 100% functional coverage.

The synthesized SVA assertions are used in block level testbench as more random seeds are executed. They are also shipped with the block RTL together to chip level simulation using Questa simulator. Note that the chip level observability is significantly improved by these whitebox assertions.

Connect Simulation and Formal Using Bugscope

Usage of formal verification suffers from two fundamental problems:

- Constraints are difficult to write and error prone. Proofs are as good as the constraints are correct. To our knowledge, there is no good way to guarantee correctness of the written constraints [4].
- Developing assertions to prove is difficult for designers.

Bugscope provides a natural solution to these two problems. Based on a Questa simulation environment, Bugscope extracts both assertions and cover properties. Note that the cover properties point to the area where simulation fails to reach. Then we give both assertions and cover properties to Questa formal verifier to prove. This methodology naturally solves the above two problems:

- Synthesized cover properties provide a good corner case target to the formal engine to reach. If they are unreachable, the constraint environment is very likely to be buggy and should be corrected;
- Bugscope supplies a high density sets of assertions for formal engine to prove. They are easier than typical manual end-to-end assertions due to its whitebox nature.

Because Bugscope can provide synthesizable set of SVAs, the integration between Bugscope and Questa Formal Verifier is painless.

New Assertion Synthesis Driven Hardware Accelerator Flow

In recent years, hardware acceleration technology has matured and been adopted by various leading chip companies. However, several fundamental issues still persist

- Testbench checking can only be applied on interface signals. Some features such as performance, whitebox behaviors are very difficult to capture at interface level.

Those types of checking are often omitted.

Therefore, the verification observability is low.

- There is no coverage measurement on the quality of verification. This can be a major hurdle for the verification team to adopt hardware acceleration.

Assertions provide a natural solution to both of the above problems. As a matter of fact, Veloce hardware accelerators can accept SVA assertions as well as SVA cover properties. The problem is which assertions or cover properties should be added to the Veloce hardware acceleration. Therefore, we propose a new Assertion Synthesis driven hardware acceleration flow (see Figure 3).

First, we use Bugscope to generate assertions and cover properties in a Questa simulation environment. At classification phase, the designers will notify which assertions and which cover properties they are willing to put into the hardware acceleration environment. Second, these assertions and cover properties are filtered through Questa Formal Verifier. If an assertion is proven to be true, it will be removed from the list because it will never catch bugs as long as RTL doesn't change. Similarly, if a cover property is proven unreachable, it will be removed from the list because it will never be reachable as long as RTL doesn't change. Note that this step is important because the resource on Veloce hardware is limited. Table 1 shows the effectiveness of the Questa Formal Verifier to help reduce the number of redundant properties. Finally, the left assertions and cover properties are integrated into Veloce Hardware Accelerator to improve the observability.

CONCLUSION

This article introduces a new Quest/Veloce verification methodology based on Assertion Synthesis technology. This methodology automates the assertion based verification and solves the observability problem in the SOC verification. A number of customers have used this methodology successfully and have found bugs in their designs [5].

REFERENCE

- [1] Harry D. Foster, Adam C. Krolnik, David J. Lacey, "Assertion-Based Design", Kluwer Academic Publishers, 2nd edition, 2004.
- [2] Yunshan Zhu, Yuan Lu, "Assertion Synthesis: Enabling Assertion-Based Verification For Simulation, Formal and Emulation Flows", Whitepaper, <http://www.nextopsoftware.com>.
- [3] "Transaction-based Simulation Acceleration Software - TestBench Xpress", <http://www.mentor.com/products/fv/emulation-systems/veloce/testbench-xpress>
- [4] Alan J. Hu, Masahiro Fujita, and Chris Wilson, "Formal Verification of the HAL S1 System Cache Coherence Protocol," IEEE International Conference on Computer Design (ICCD), pp.438--444, 1997.
- [5] Jing Li, Nantian Qian, Yuan Lu, "Linking Multiple Verification Flows Using Automatically Generated Assertions", DVCon, 2011.

Figure 3 Assertion Synthesis Driven Hardware Acceleration Flow



Table 1 use Questa Formal Verified to Reduce # of Assertions on Veloce

Block Name	#FFs	#Assertions	#Proven Assertions	#Checkin Assertions
BLOCK1	10330	128	29	99
BLOCK2	20783	101	12	89
BLOCK3	169518	68	37	31
BLOCK4	69566	65	7	58
BLOCK5	270197	362	85	277