World Class Verilog, SystemVerilog & OVM/UVM Training

Are Advanced Verification Methodologies Required to Test FPGA Designs?

Clifford E. Cummings
Sunburst Design, Inc.
www.sunburst-design.com
cliffc@sunburst-design.com

Abstract

Todays FPGAs are the size of ASICs from just a few years ago and the older techniques of only testing an FPGA in the lab are inadequate and irresponsible. Modern verification methodologies, like UVM (the SystemVerilog Universal Verification Methodology) are required to functionally verify modern FPGA designs.

Introduction

Are advanced verification methodologies, like UVM (the SystemVerilog Universal Verification Methodology) required to verify todays FPGA designs? The answer is yes.

Todays FPGAs are the size of ASICs from just a few years ago and the older techniques of only testing an FPGA in the lab are inadequate and irresponsible.

There is still a prevailing notion in the FPGA design community that FPGAs do not have to be extensively verified through simulation, that FPGAs can be tested in the lab and corrections made when errors are found, but that simply is not true for multiple reasons.

Issues related to omitting functional simulation

So what are potential consequences of omitting functional simulation of a design?

(1) Unless you run code coverage, you don't even know what code has not been tested. You don't even know if the untested code works. Running FPGAs in a lab cannot identify what code has not been tested.

(2) Unless you run functional coverage, you don't know which features have not been tested (requires a test plan). Running FPGAs in a lab cannot reliably identify which features have not been tested.

(3) Directed tests (and in-lab testing) will find the bugs you are looking for. Random testing will find the bugs you were not looking for.

(4) Binding SystemVerilog Assertions (SVA) to a simulated design can reduce debug time by more than 50% over non-assertion testing or in-lab testing. SVA improves observability into a design making it easier to identify and correct design flaws.

(5) If a customer finds a bug, the cost to fix the bug can be much greater than the cost to functionally verify the design before shipping.

(6) If a customer finds a bug, the impact to your company's reputation could far exceed the cost to verify the design before shipping.

(7) If a bug is on a mission-critical application or a failing product jeopardizes the life of the user, there could be significant legal ramifications when it is proven that well known industry and commonly used standard methodologies (such as UVM and SVA) were in available that could have identified the flaws before the product was shipped.

What is Functional Coverage?

Many engineers do not know what functional coverage is. A quick explanation follows.

Best known practices for modern design verification requires three general steps:

Functional Simulation - simulation proves that the design is functionally correct, but it does not prove that the entire design has been tested or that all of the features have been tested.

Code Coverage - used with functional simulation, this proves that all of the code has been touched and therefore all of the code has been tested, but it does not prove that the desired features are working. 100% code coverage tells you that all of the code has been tested but again does not tell you if you have tested all of the features. Code coverage does not answer the question, "are we done testing yet?" Code coverage only tells you which blocks of code have not even been tested so you do not know if that code is working or not. This step is largely automatic and simply requires a few simulator-specific switches to be turned on during simulation.

Functional Coverage - used with functional simulation, this proves that all of the features of a design have been tested without requiring an engineer to reexamine waveforms after each simulation. Functional coverage requires a test plan, which serves as a contract between the verification engineers, the design engineers and any other interested parties such as architects or marketing teams that may have identified required product features. The verification engineer presents the test plan to the designers and others, and asks, "based on my understanding of the design specification, if I test these features, under these conditions, with random values weighted towards these corner cases, do we all agree that the required design features have been adequately tested?" Once the test plan has been accepted by all interested parties, the verification engineer sets out to test the design and prove that all features have been 100% tested using SystemVerilog functional coverage.

Used together, code coverage shows that all of the code has been tested, functional coverage proves that all of the identified features were tested, and simulation proves that all of the tested code and features were working correctly. This is what is required to pronounce that, "yes, we are done testing."

Prior to the addition of functional coverage capabilities, engineers either had to manually examine simulation waveforms to identify that features were working or write their own Verilog behavioral code to capture working features during simulation. Both were tedious and time-consuming tasks. The addition of SystemVerilog functional coverage language features greatly simplified the task of automating the logging of working design features during a simulation.

It should be noted that functional coverage test plans often evolve as a design is being tested. While testing a design, engineers often realize that certain important features were not identified and included in the original test plan and that the plan needs to be amended to add the additional important features.

Actual FPGA Design Flaw Experience

Many years ago, I worked on project where an FPGA was designed by a contract engineer, which was not functionally verified using simulation. The product worked fine in the prototypes and worked fine for the first six months that the product was in production. After a new batch of FPGAs arrived on the production line, the product started to fail due to slight differences in device timing. The contract engineer was gone and I knew something about the design, so I was tasked with finding and fixing the problem.

You have never truly been in the critical path of a project until you have shut down the manufacturing line of a product that is contributing revenue to the company. I hope I never have that experience or pressure ever again. To make matters worse, the contract designer had not archived the very latest copy of the design files, so I had to first find the last modifications and update the design before I could run the simulations, identify the race condition and fix the problem. The product line was shut down for almost two weeks while I fixed the problem and the project I was working on was put on hold during that same two-week period. That was a small FPGA by todays standards.

Summary

In summary, with existing technologies and methodologies, it is simply irresponsible to release any product that has not been fully tested using proven industry-standard techniques. Releasing such products exposes companies to significant legal liability and loss of reputation. No company should take that risk.