Design Verification Engineer - the expected skill set
Yuri Panchul, 2023-04-15

A Design Verification Engineer is essentially a specialized software engineer who understands how hardware blocks function and uses specific methodologies to verify them. He needs to possess the following skills:

1. The ability to understand an architectural specification for a hardware block. The architectural specification is a document, typically between 30-100 pages long, describing the block signal interfaces and the block functionality on the transaction level.

   “Transaction level” means that the block functionality is described as a sequence of transaction processing events rather than on a clock cycle-by-cycle basis. An example of a transaction could be a network packet or a memory read or write request.

2. The ability to create a test plan. This plan should enumerate different scenarios of block usage and specify how these cases should be verified: by checking the outputs of the hardware module against a software model during simulation or other methods.

3. Understanding of the basics of digital design on RTL, Register Transfer Level: the concepts of signals, clock, reset, combinational and sequential logic, finite state machines, the idea of pipelining, flow control, valid/ready and credit-based interfaces, on-chip buses like AMBA AXI, types of memory devices (single, dual and pseudo-dual-port SRAM memories) and clock domain crossing.

4. Understand SystemVerilog as a language, including bit manipulations, data structures (queues, associative arrays), always/initial/final blocks, fork/join threads, module hierarchy, tasks/functions and object-oriented programming. OOP in SystemVerilog is used in widespread libraries, most notably Universal Verification Methodology (UVM) library.

   A helpful reference material is SystemVerilog for Verification: A Guide to Learning the Testbench Language Features by Chris Spear and Greg Tumbush, 2014.

5. Understand the mechanics of event-based simulation in SystemVerilog; blocking, non-blocking and concurrent assignments; delays, delta delays, named events and race conditions.

   A reference material: Logic Design and Verification Using SystemVerilog by Donald Thomas, 2016.

6. Ability to write self-checking testbenches (verification environments) that utilize scoreboards - a specially organized code used to compare the responses of the design under test against a transaction model.

7. Ability to debug the design together with a testbench using a debug environment such as Siemens EDA Questa, Synopsys Verdi or Cadence SimVision. Analyzing waveforms, tracing signals drivers and loads, counting events, and writing scripts in Tcl to automate simple debug tasks.

8. Since some models can be written in C/C++ or other languages, a DV engineer should understand or at least be aware of SystemVerilog Direct Programming Interface (DPI).

9. A DV engineer should understand the functional coverage-based constrained random verification methodology, a modern technique to control the testing quality. Should be able to write SystemVerilog classes with random fields and constraints, and understand how the constraint solver works. Should be able to write functional covergroups and interprets the coverage reports produced during simulation.

10. Understand the concept of temporal logic and SystemVerilog Assertions (SVA), a sub-language of SystemVerilog used to define sequences (signal changes over multiple clock cycles) and properties (statements about sequences). These mechanisms are used to find bugs during simulation, to check functional coverage (cover properties) and to aid in formal verification.

11. Exposure to the Universal Verification Methodology (UVM) library, an object-oriented library that helps to build verification environments.

    A helpful short tutorial book: Getting Started with UVM: A Beginner's Guide by Vanessa R. Cooper, 2013.

12. An understanding of how to verify the specific cases of the design: CPUs, using instruction set simulators as golden models; network routers, building scoreboards for the network packets; system on chips (SoCs), by co-simulating multiple components.

13. An understanding of formal verification techniques, such as proving SVA properties or sequential equivalence, is considered a plus for a DV engineer.

14. Understanding the concept of hardware emulation (Siemens Veloce, Synopsys ZeBu) and experience in debugging a project on an emulator is also considered a plus for a DV engineer.

15. Since ASIC designers and DV engineers almost always work in Linux environment, a DV engineer should have solid Unix/Linux background. A DV engineer should be proficient with command line and regular expressions, Bash scripting and makefiles, text processing tools like sed/awk and/or Python scripting. Such skills are needed to run the simulation jobs and extract data from simulation reports.

16. Other skills needed for a DV engineer productivity: Git version-control system, using continuous integration pipelines, working with bug tracking software, and running the simulation regressions on a server farm.

Yuri Panchul, 2023-04-15