A Layered Approach

CHAPTER 4

It is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail.

Abraham Maslow

For longer than we know, humans have organized themselves into layers. From the family and tribe all the way up to national governments, we have created roles and responsibilities. Closer to the hardware domain, both VHDL and Verilog also use a layering concept, employing entities or modules to break up a task. The software domain uses the related concepts of procedures (*methods*) and data structures (*classes*). A reason we humans make layers, with associated roles and responsibilities, is to simplify our lives.

This chapter looks at how using layers can organize the task of verifying a chip. We look at a generic chip, albeit one with a "System-on-a-Chip" bias, and come up with a set of standard, well-defined layers, roles, and responsibilities. We leave this chapter with definitions of standard verification layers and detailed diagrams of functional "boxes" and how they are interconnected. Part II of this handbook will show a fully implemented C++ environment that uses this approach. Part III will talk

about general C++ techniques for implementing these classes and connections. These techniques, applicable to most of the languages used for verification, express the reasoning behind the layered approach discussed below.

Overview

Throughout this chapter, little distinction is made among architecture, design, and coding. This is because these activities are interrelated, and occur at most stages in a project. Also, even with the initial architectural efforts, you should have a plausible implementation in mind; otherwise, the architecture may create problems when you are coding.

At many layers, verification environments tend to have the same set of problems. The essence of this chapter is to show how these common problems can lead to common solutions. By reusing solutions, the team can be more productive.

Specifically, this chapter covers the following topics:

- The importance of code layers, roles, and responsibilities
- How to go from a whiteboard verification system to classes and interconnects, using a standard framework
- Some common components, roles, and responsibilities of a verification system

There are many successful hardware products. Because success demands more success, the hardware produced in the next revision of a product will be more complex than the current version. In addition, the sales staff wants the product in the shortest possible time. The three competing factors of quality, functionality, and time to market create stress on the verification team. You are expected to produce more in less time—*and* with increased quality.

So how do you do that? You could add members to your team. While it is certainly true that there is an appropriate number of people for every task, adding people creates several issues. One is the need for increased communication; adding a team member increases the need for each team member to interconnect with the rest of the team, decreasing productivity.