Introduction

CHAPTER 1

Coding is a human endeavor. Forget that and all is lost.

Bjarne Stroustrup, father of C++

There are several books about hardware verification, so what makes this handbook different? Put simply, this handbook is meant to be useful in your day-to-day work. The authors are like you, cube dwellers, with battle scars from developing chips. We must cope with impossible schedules, a shortage of people to do the work, and constantly mutating hardware specifications.

We subtitled this book *A Practitioner's Handbook* because it contains real-world code examples and techniques. Sure, we talk about programming theory, but the theme of this book is writing code. We focus mainly on object-oriented programming (OOP) techniques and coding in C++. We back this up with a CD-ROM containing working, open-source Verification Intellectual Property (VIP), scripts, and several complete test systems.

We cover the following topics:

- C++ as a verification language
- The evolution of OOP, C++, and verification
- How to use OOP to build a flexible and adaptable verification system
- How to use specific OOP techniques to make verification code both simpler and more adaptable, with reference to actual situations (both good and bad) that the authors have encountered
- Useful C++ code, both as snippets, complete examples, and code libraries—all available as open source

This handbook is divided into four major sections:

- *Part I* provides an overview of OOP concepts, then walks through transforming a block-level view of a typical verification system into code and classes.
- *Part II* describes two free, open-source code libraries that can serve as a basis for a verification system—or as inspiration for your own environment. The first, called Teal, is a C++-to-HDL (hardware description language) interface. The second, called Truss, is a complete verification system framework. Both are available as open source and are included on the companion CD-ROM.
- *Part III* describes how to use OOP to make your team as productive as possible, how to communicate design intent better, and how to benefit from "lessons learned" in the software world.
- *Part IV* describes several complete real-world examples that illustrate the techniques described in the earlier parts of this book. In these examples we build complete verification environments with makefiles, scripts, and tests. These examples can serve as starting points for your own environment.

For the curious, each of the chapters in Part I and Part III ends with a section called "For Further Reading," which recommends relevant landmark papers and books from the software domain.[1]

[1] The references in these sections, though not academically rigorous, should be sufficient to help you find the most recent versions of these works on the Internet.