

# Chapter 1

## Verification Guidelines

*“Some believed we lacked the programming language to describe your perfect world...”*  
*(The Matrix, 1999)*

Imagine that you are given the job of building a house for someone. Where should you begin? Do you start by choosing doors and windows, picking out paint and carpet colors, or selecting bathroom fixtures? Of course not! First you must consider how the owners will use the space, and their budget, so you can decide what type of house to build. Questions you should consider are: Do they enjoy cooking and want a high-end kitchen, or will they prefer watching movies in their home theater room and eating takeout pizza? Do they want a home office or an extra bedroom? Or does their budget limit them to a more modest house?

Before you start to learn details of the SystemVerilog language, you need to understand how you plan to verify your particular design and how this influences the testbench structure. Just as all houses have kitchens, bedrooms, and bathrooms, all testbenches share some common structure of stimulus generation and response checking. This chapter introduces a set of guidelines and coding styles for designing and constructing a testbench that meets your particular needs. These techniques use some of the same concepts that are shown in the *Verification Methodology Manual for SystemVerilog* (VMM), Bergeron et al. (2006), but without the base classes. Other methodologies such as UVM and OVM share the same concepts.

The most important principle you can learn as a verification engineer is: “Bugs are good.” Don’t shy away from finding the next bug, do not hesitate to ring a bell each time you uncover one, and furthermore, always keep track of the details of each bug found. The entire project team assumes there are bugs in the design, so each bug