Chapter 11

A Complete SystemVerilog Testbench

This chapter applies the many concepts you have learned about SystemVerilog features to verify a design. The testbench creates constrained random stimulus and gathers functional coverage. It is structured according to the guidelines from Chap. 8 and so you can inject new behavior without modifying the lower-level blocks.

The design is an ATM switch that was shown in Sutherland et al. (2006), who based his SystemVerilog description on an example from Janick Bergeron's Verification Guild. Sutherland took the original Verilog design and used SystemVerilog design features to create a switch that can be configured from $4 \times 4$ to $16 \times 16$. The testbench in the original example creates ATM cells using $\$urandom$, overwrites certain fields with ID values, sends them through the device, and then checks that the same values were received.

The entire example, with the testbench and ATM switch, is available for download at http://chris.spear.net/systemverilog. This chapter shows just the testbench code.

11.1 Design Blocks

The overall connection between the design and testbench, shown in Figure 11-1, follows the pattern shown in Chap. 4.
Figure 11-1 The testbench – design environment

The top level of the design is called *squat*, as shown in Figure 11-2. The module has 1..*N* Utopia Rx interfaces that are sending UNI-formatted cells. Inside the DUT, cells are stored, converted to NNI format, and forwarded to the Tx interfaces. The forwarding is done according to a lookup table that is addressed with the VPI field of the incoming cell. The table is programmed through the management interface.

Figure 11-2 Block diagram for the *squat* design

The top level module in Sample 11.1 defines arrays of interfaces for the Rx and Tx ports.