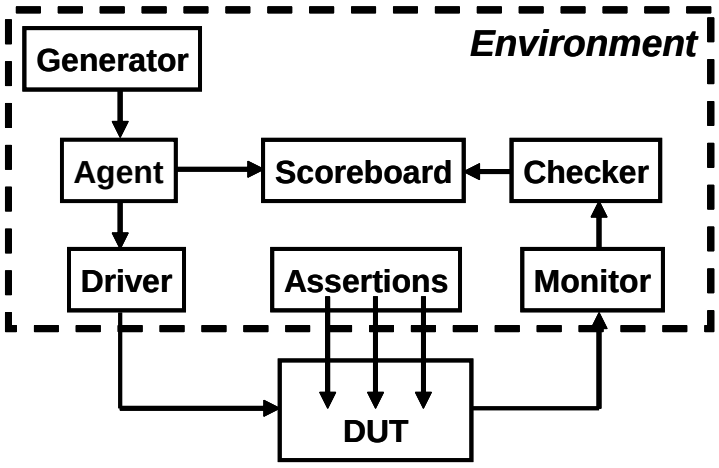
Chapter 7

Threads and Interprocess Communication

7.1 Introduction

In real hardware, the sequential logic is activated on clock edges, while combinational logic is constantly changing when any inputs change. All this parallel activity is simulated in Verilog RTL using **`initial`** and **`always`** blocks, plus the occasional gate and continuous assignment statement. To stimulate and check these blocks, your testbench uses many threads of execution, all running in parallel. Most blocks in your testbench environment are modeled with a transactor and run in their own thread.

Figure 7-1 Testbench environment blocks



The SystemVerilog scheduler is the traffic cop that chooses which thread runs next. You can use the techniques in this chapter to control the threads and thus your testbench.

Each of these threads communicates with its neighbors. In Figure 7-1, the generator passes the stimulus to the agent. The environment class needs to know when the generator completes and then tell the rest of the testbench threads to terminate. This is done with interprocess communication constructs such as the standard Verilog events, event control and **`wait`** constructs, and the SystemVerilog mailboxes and semaphores.[10]