function correctly) and timing verification (that the circuit meets timing constraints). We outlined in Section 1.5 how tools perform static timing analysis to verify timing constraints. Here, we will discuss functional verification using Verilog models, expanding on the ideas introduced in Section 2.4 relating to verification of combinational circuits.

When verifying a combinational circuit, we saw that we need to wait for some time after applying a test case to the circuit's inputs before checking the circuit's outputs, to allow for the propagation delay of the circuit. Similarly, when verifying a sequential circuit, we need to take account of the fact that operations take one or more clock cycles to complete. We need to ensure that the procedural block that checks the output is synchronized with the stimulus block, and knows how many clock cycles after application of a test case to wait before checking the output. If all operations complete in the same number of cycles, and only one operation takes place at a time, this is relatively straightforward. On the other hand, if operations take varying numbers of cycles to complete, the checker needs to check both that the operation completes at the correct time and that the correct result is produced. If multiple operations can take place concurrently, for example, if the datapath is a pipeline, the checker needs to ensure that all operations that start also complete, and that no spurious results are produced.

Developing testbench models for complex sequential circuits can itself become a complex endeavor. We will discuss some of the general techniques that can be used in Chapter 10. For now, we will illustrate a simulation-based approach for verifying circuits that we introduced in previous examples.

EXAMPLE 4.21 Develop a testbench model for the sequential multiplier of Example 4.14. Verify that the result computed by the multiplier is the same (within the limits of the precision of the operands) as that produced using multiplication with the built-in Verilog type real.

SOLUTION The testbench has no external connections, and so the module definition is

```
`timescale 1ns/1ns

module multiplier_testbench;

  parameter t_c = 50;

  reg               clk, reset;
  reg               input_rdy;
```

(continued)