```
3   endsequence
4
5   property p1 (q, r, s);
6     @(posedge ck) s1(q, r, s);
7   endproperty
8
9   assert property (p1 (q, r, s)) else $error("oops");
```

In this case, we have specified the positive edge of the clock and we have used variables q, r, and s in all places to help keep the explanation clearer. Given the description of what an assertion does at each clock edge (Section 9.2.3), at every posedge of clock ck, the property tries to assert sequence s1 (line 6 above). Of course, since q isn't always TRUE at each clock tick, there will be vacuous successes that will cause the assertion to wait for the next clock tick.

```
1   module assertQRS;
2     bit       q=0, r=0, s=0;
3     bit       ck=0;
4
5     always #5 ck = ~ck;
6
7     initial begin
8       $monitor($stime,,,
9         "ck=%b, q=%b, r=%b, s=%b",
10          ck, q, r, s);
11      q <= #4 1;
12      q <= #6 0;
13      r <= #14 1;
14      r <= #16 0;
15      q <= #14 1;
16      q <= #16 0;
17      r <= #24 1;
18      r <= #26 0;
19      s <= #44 1;
20      s <= #46 0;
21      #56 $finish;
22    end
23
24    sequence s2(r, s);
25      (r ##3 s);
26    endsequence
27
28    property checkQRS (q, r, s);
29      @(posedge ck)  q |=> s2(r, s);
30    endproperty
31
32  P1a: assert property (p2(q, r, s))
33           $display("%d Yes!", $stime);
34           else $error("%d oops", $stime);
35  ...
```

Example 9.4 — Simulation of checkQRS

Most assertions of this type are started with an *implication construct* that only starts an assertion if a specific condition is TRUE. In this way, a vacuous success is avoided. An implication takes one of two forms:

```
1  i |=> j;     // i TRUE now implies that j will be TRUE at
                the next clock tick
2  i |-> j;     // i TRUE now implies that j is TRUE at the
                current clock tick (now)
```

where, in each case here, i is an expression that evaluates to TRUE or FALSE, and j is typically a sequence. If i is TRUE, the property tries to start recognizing the sequence. Implications such as this are only used within a property specification (property..endproperty). The difference between the two forms of implication is when the sequence is to start. With the first form (|=>), the first step of the sequence is checked for at the following clock tick. With the second form (|->), the first step of the sequence should be TRUE now, in the current time.

Our sequence and assertion can now be rewritten using the |=> implication construct as shown on lines 24-30 of Example 9.4.

Two changes have been made to the sequence and property statement. First, the sequence has been changed to include all but the first variable: thus instead of the sequence being (q ##1 r ##3 s), it will now be (r ##3 s) as shown on line 25. Second, the q will now be used as part of an implication in the property on line 29. The way to read line 29 is: at the positive edge of ck, if q is TRUE, then begin asserting sequence s2 at the next negative clock edge. This still results in the same sequence; the ##1 in the original sequence is now specified as part of the implication.