...). Following the initial block on lines 8-15, q become 0 at time 0, changes to 1 at time 15 (on the second positive edge), and then changes to 0 at time 55.

9.2.2 Properties and Sequences

The property shown above used the variable names q, r, and s which are defined as bit variables in the example. Properties can also be specified with formal parameters. Then the assertion statement can provide the actual parameters to be used. Consider the following equivalent code for Example 9.2.

```
1   property a1b3c (a, b, c);
2     @(posedge ck) a ##1 b##3 c;
3   endproperty
4
5   assert property (a1b3c (q, r, s)) else $error("oops");
```

The only difference here is that the property uses formal parameters. Then when the property is asserted on line 5, variables q, r, and s are substituted for a, b, and c respectively.

Sequences of values in a design are important enough that they can be defined and given a name using the keyword "sequence." In the case of the sequence in the above property, it could have been defined as:

```
1   sequence abxxc (a, b, c);
2     a ##1 b ##3 c;
3   endsequence
```

Then a property could have been defined to use that sequence:

```
1   property checkQRS (q, r, s);
2     @(posedge ck) abxxc (q, r, s);
3   endproperty
```

Note here that both the property and sequence use formal parameters enabling for maximum flexibility in their use.

It's possible that sequence abxxc is only part of a sequence to be checked. For instance, if variable d should be TRUE two ticks after abxxc, we could check for the property:

```
1   property checkQRSD (q, r, s, d);
2     @(posedge ck) abxxc (q, r, s) ##2 d;
3   endproperty
```

9.2.3 How to Think About the Execution of Assertions

We've stepped through an example based on Figure 9.1. Now let's formalize how it works. A concurrent assertion will try to recognize the start of its sequence at every tick of the clock. Consider property checkQRS above and the following assert statement:

```
1   assert property (checkQRS (q, r, s)) $display("Yesssss!") else $error("Noooo!");
```

This statement asserts the specified property with the actual parameters given. Based on this activation, the property will try to start and match its sequence on every clock edge, as specified in the property and its sequence. One of five things can happen at each clock tick: