9.3.2 A property and sequences for the protocol

The basic property for this bus protocol would start out as:

```
property x;
   @(negedge ck) start |=> //some sequence
endproperty
```

The point being that the signal start is what sets off the protocol and we can use it to start up certain sequences to check various features of the protocol. These sequences can either be defined separately as sequences and then invoked on line 2, as was done in the earlier sections, or the sequence can be written on line 2 as part of the property. To make the discussion easier to follow, we'll write the sequences as part of the property.

First, let's assume that there is no delay state (S4 in Figure 9.5). The new version of the waveforms are shown in Figure 9.6. Now, read and write cycles take three clock ticks. A simple assertion to check that a read bus cycle starts and ends in three ticks of the clock would be:

```
1  property readIs3ticks;
2     @(negedge ck) start |=> read ##1 dataValid;
3  endproperty
```

This property checks that if start is valid now (clock tick B), read will be valid at the next clock tick (as specified by the |=> implication) and dataValid will be asserted one tick later.

SystemVerilog provides some useful system calls for determining if certain values are unknown. The task $isunknown(y) returns TRUE if the expression y is unknown (i.e., it contains either x or z values). Since the address bus is normally high impedance, we could check to make sure that the address lines are driven when start is asserted:

```
1  property addressBusDriven;
2     @(negedge ck) start |-> ~$isunknown(address);
3  endproperty
```

In this example a different form of implication is used. The earlier form used (|=>) specified to look for the next element of the sequence at the next clock tick, this form (|->) says to look now — when start is TRUE. The way to read this is that when start is TRUE, the address bus should be not-unknown right now. It's a standard implication form; you could read it as "start implies address bus not-unknown."

The readIs3ticks property tested to see if a read operation was done correctly. That is, after the start signal was seen, it watched for read being asserted at the next tick. If the operation on the bus was a write, then this property would fail because read wouldn't be asserted. Again, when testing a design, you don't want your properties failing even though there is no problem.

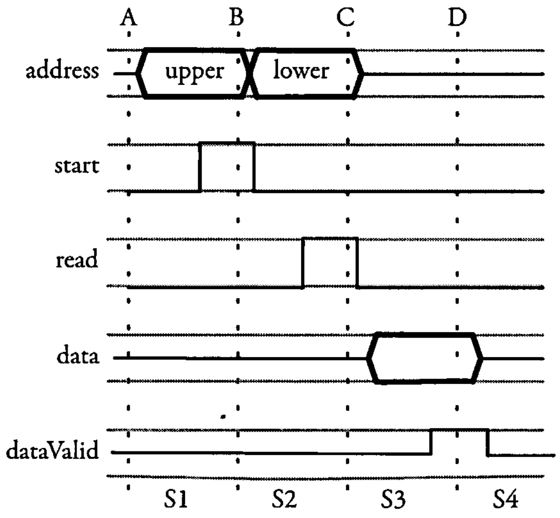

Figure 9.6 — SimpleBus Read Protocol Without Delays

One approach to testing this would be to just test for dataValid at the third tick. This is shown here: