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

This sequence says that when start is asserted, then at the next tick begin following the sequence. The sequence says to wait one clock tick and check for dataValid. The property could have been written equivalently as:

```
1  property ckDataValidOnlyAlt;
2    @(negedge ck) start |-> ##2 dataValid;
3  endproperty
```

In this case, the implication (|->) says to start following the sequence in the same tick time as when start is TRUE. In either case, the effect is to wait for 2 clock ticks before checking for dataValid (i.e., 3 ticks in all).

With a slightly more complex firing condition for the implication, the notion of a read could be accommodated. Consider the following property:

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

In this property, the condition for starting is that start is TRUE followed by read at the next clock tick. If that occurs, that means that a read is in progress and dataValid should be TRUE at the next tick.

The complementary check for a write would be:

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

The only difference being that ckWrite watches for read not being asserted as part of the starting condition.

9.3.3 Sequence Operators

Now we return to the full protocol specification (Figure 9.5) where there is an unspecified time delay between the start signal and the dataValid. We'll define this to be as small as 2 ticks and as large as 7 ticks for a write, and as small as 2 and as large as 10 ticks for a read; all these delays are from start. Thus the shortest time is the time checked for above where dataValid comes right after read in a read cycle (2 ticks from start).

To specify a range of tick counts in a sequence, the construct ##[n:m] is used; n is the minimum and m is the maximum. Considering only the read cycle, the property would be:

```
1  property ckReadRange;
2    @(negedge ck) (start ##1 read) |-> read ##[1:9] dataValid;
3  endproperty
```

This sequence says that when we see start at the current tick, we should see read at the next tick. This is the antecedent that starts the implication. From there, read should be TRUE in the current time and dataValid between one and nine ticks in the future.