- *or* — two sequences begin at the same clock tick. As soon as one succeeds, the assertion succeeds. If neither succeed, then the assertion fails.
- *and* — two sequences begin at the same clock tick. Both sequences must succeed although their ending times may be different.
- *intersect* — like and (above) but with the further restriction that both sequences end at the same time.
- *throughout* — the expression on the left is to remain TRUE throughout the sequence on its right.
- *within* — specified as "s1 within s2". s1 must start no earlier than the start of s2 and must end no later than the end of s2.
- *first_match* — whenever sequences are specified with the above operators and also have range repetitions, there can be several matches at different times. This reports the first and the others are discarded.

9.3.4 Signal/Expression Repetitions in Sequences

In some protocols there is a signal or a sequence of signals that can repeat.

Consider the ckOnlyOneStartAlt property (above) which states that once the sequence is started, start is FALSE until the tick after dataValid. This was specified using a range on the ## delay operator. Another approach would have been to specify it as a range of iterations of the start signal:

```
property ckOnlyOneStartRepetitions;
   @(negedge ck) start |=> ~start [*1:8] ##1 (dataValid & ~start);
endproperty
```

This sequence states that one tick after start is seen, there will be between 1 and 8 consecutive ~starts followed by a dataValid AND a ~start at the same tick. There are several things to note here:

- The consecutive range repetition operator, written here as [*1:8], specifies that the element to its left, which is ~start, will repeat in consecutive clock ticks between 1 and 8 times (inclusive). The construct is left associative.
- In the last tick, the one that breaks the repetition loop of watching ~start, dataValid and ~start will both be TRUE. This assures that when dataValid occurs that ~start will still be TRUE. This is because there can't be another start until one tick after dataValid. If the ~start condition was left out of the dataValid expression, a new start could have occurred while dataValid was asserted — that's incorrect in this protocol.

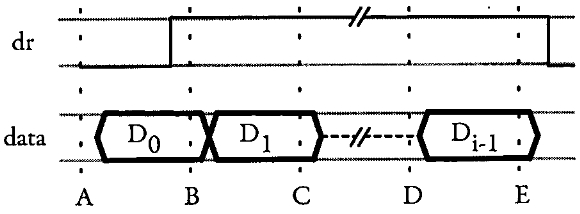

Figure 9.7 — dr Indicating When a Block of i Data Bytes Are Being Sent

Consider another protocol specification, this one shown in Figure 9.7. In this protocol, a series of data bytes is placed on the data lines by a sender at consecutive clock ticks. A data ready (dr) signal is asserted by the sender whenever there is data on the line. As long as dr is asserted, the re-