The Implication Process

The tasks of the implication process (represented by the routine *Imply_and_check* in Figure 6.10) are

- Compute all values that can be uniquely determined by implication.
- Check for consistency and assign values.
- Maintain the *D-frontier* and the *J-frontier*.

We can view the implication process as a modified zero-delay simulation procedure. As in simulation, we start with some values to be assigned; these assignments may determine (imply) new values, and so on, until no more values are generated. All values to be assigned are processed via an *assignment queue* similar to the event queue used in simulation. Unlike simulation, where values only propagate forward (toward POs), here values may also propagate backward (toward PIs). An entry in the assignment queue has the form (l, v', *direction*), where v' is the value to be assigned to line l and *direction* $\in$ {*backward, forward*}. To generate a test for the fault l *s-a*-1, the initial two entries in the assignment queue are (l, 0, *backward*) and (l, $\overline{D}$, *forward*).

Imply_and_check retrieves in turn every entry in the assignment queue. The value v' to be assigned to l is first checked for consistency with the current value v of l (all values are initialized to x). An inconsistency is detected if $v \neq x$ and $v \neq v'$. (An exception is allowed for the faulty line, which gets a binary value to be propagated backward and an error value to be propagated forward.) A consistent value is assigned, then it is further processed according to its *direction*.

Backward propagation of values is illustrated in Figure 6.14. The right side of the figure shows the effects of the assignments made on the left side. An arrow next to a logic value shows the direction in which that value propagates. The assignment a=0 in Figure 6.14(c) causes a to be added to the *J-frontier*. Figure 6.14(d) shows how backward propagation on a fanout branch may induce forward propagation on other fanout branches of the same stem.

Similarly, Figures 6.15 and 6.16 illustrate forward propagation of values. Note in Figure 6.15(d) how forward propagation on a gate input may induce backward propagation on another input of the same gate.

If after all values have been propagated, the *D-frontier* contains only one entry — say, a — then the only way to propagate the error is through gate a. Figure 6.17 illustrates the implications resulting from this situation, referred to as *unique D-drive*.