Unfortunately, not all negative edge-triggered flops can be treated as buffers, such as neg_flop1 and neg_flop2 in Figure 10. The solutions can be any one of the following:

- Take neg_flop1 and neg_flop2 out of the scan chain, which means combinational logic blocks L1 and L2 are not testable, reducing coverage.
- Put neg_flop1 and neg_flop2 into a separate scan chain and model them as negative edge-triggered flops.
- Put neg_flop1 and neg_flop2 at the beginning of the existing scan chain. The scan data can be shifted in correctly and the primary outputs strobed. But the system clock cannot be applied to both negative and positive edge-triggered flops at the same time. Therefore, the test tool has to be run twice, once for each type of flops. Two sets of test vectors will be generated and run together during chip fabrication.

DFT Rule 9: Try not to multiplex PLL test pins with design primary I/O pins to reduce pin count.

Most ASIC vendors provide their own PLL block and associated self-test that requires several primary I/O pins. Many designers like to multiplex these test pins with design primary I/O pins to reduce pin count (if performance degradation is not an issue), because PLLs can never be tested when the ASIC is functionally operating. Figure 11 shows this design practice. Faults on pin i1, s and z are not testable. One solution is to insert a scan flop between PLL block and pin io of the multiplexer. Consider DFT consequences when you make a design decision such as reducing pin count.

DFT Rule 10: Do not use bidirectional signals as pure unidirectional ones. Figure 12 shows an example of this.

DFT Rule 11: If your gate-level netlist is in Verilog and if your test tool directly uses this netlist to generate ATPG vectors, then do not use any "assign" statements in it because some test tools automatically translate each assign statement into a pseudo buffer cell in the ATPG database. This pseudo cell is seen as an element of the netlist and its "faults" are targeted by ATPG, causing the target fault list to be different from the actual design. Unfortunately, most of such faults are ATG_UNTESTABLE and it takes time to identify them. An exception to Rule 11 is the use of "assign" statements in VDD and GND definitions, such as the following:

assign VDD = 1'b1;
assign GND = 1'b0;

Some back-end process tools automatically insert the words VDD and GND into the gate-level netlist they generate. If the netlist does not have the two "assign" statements above, then it cannot be simulated.

Now we have finished discussing important DFT rules. Of course, a complete list of DFT rules is beyond the scope of this discussion, but those that we have discussed are common in real-world designs. If you follow them, you will have a very easy time with scan testing; if you don’t, you won’t. Some of the problems discussed so far cannot be solved unless you change the design. Design change during or after synthesis is always costly in time and resources.

Untestable faults

Now you know the most common DFT problems seen in real-world designs and how to perform fault-coverage analysis to identify the reasons for untestable faults.

The next logical question has to be what we can do about those untestable faults in order to raise fault coverage to beyond 95 percent. For each uncovered fault, you can do one of the following:

- Treat it as Detected Fault, such as those on qn pins.