13 COMMON ERRORS

Before moving on to chapter 14, which covers debugging, it may save some time to look at some of the common modeling errors and how to correct them.

MISMATCHED PORTS

One of the first things to double-check in a simulation is the port connections. Mismatched ports are extremely common. They are most common in designs developed by more than one person. Port mismatches are differences between a module's declaration and its connections when instantiated.

There are three types of mismatched port problems:

- incorrect number of ports
- incorrect size of ports
- incorrect order of ports

Verilog warns you if there are too many or too few port connections. Verilog also warns you if the port sizes between the declaration and instantiation are mismatched. There is no single clear warning if the order of ports is mismatched.

Mismatched port order is the most common of the port connection errors. Again, there may be no warning at all if the incorrect hookup results in a legal circuit. Mismatched port order may also result in any number of Verilog error or warning messages. If you switch the order of two signals that are of different sizes, you might get a port size mismatch warning message. If you connect an output to a register, you may get an error message about an illegal declaration.

The best way to avoid mismatched ports is to adopt some simple rules for port order. For simple modules with one output (such as a mux), use the same rules as apply to the Verilog built-in primitives: state the output first, then the inputs, followed by the control input. For an adder, you might code the design by starting with the most significant output and proceed to the least significant input, for example, *carry_out*, *sum*, *a*, *b*, and *carry_in*. For more complex logic, make up rules of your own: Are clocks described first or last? Do inputs come first, or do outputs? Should ports be in alphabetical order?

Even having a good set of rules for port order does not always prevent mismatched port order. Whenever you create an instance of a module, compare the instance against its declaration.

MISSING OR INCORRECT DECLARATIONS

Missing declarations can generate various of warning and error messages. A missing register declaration might generate a message about an illegal assignment or missing declaration. A missing width declaration can generate a warning about mismatched port sizes, illegal use of bit selects or part selects, and other messages.

Missing Registers

In behavioral modeling, most of the data objects you use in the model will be registers. In Verilog, you must declare a register before you use it. There are two places where you might forget a register declaration: Your outputs may need to be declared as *reg*s; or internal storage and temporary variables may need to be declared as *reg*s.

Look at every procedural assignment. The object on the left-hand side of the assignment must be declared as a *reg* (or possibly as an *integer*, *real*, or *time*). The error messages for a missing register declaration will point to the procedural