Sec 12.8 Port Connection Rules

Replace with:

12.8 Port Connections Rules

SystemVerilog extends Verilog port connections by making all variable data types available to pass through ports. It does this by allowing both sides of a port connection to have the same compatible data type, and by allowing continuous assignments to variables. It also creates a new type of port qualifier, **ref**, to allow shared variable behavior across a port by passing a hierarchical reference.

12.8.1 Port Connection Rules for variables

If a port declaration has a *variable* data type, then its direction controls how it can be connected when instantiated, as follows:

- An **`input`** port may be connected to any expression of a compatible data type. A continuous assignment is implied when a variable is connected to an **`input`** port declaration. Assignments to variable declared as an **`input`** port are illegal. If left unconnected, the port has the default initial value corresponding to the data type.
- An **`output`** port may be connected to a variable (or a concatenation) of a compatible data type. A continuous assignment is implied when a variable is connected the **`output`** port of an instance. Procedural or continuous assignments to a variable connected to the **`output`** port of an instance are illegal.
- An **`output`** port may be connected to a net (or a concatenation) of a compatible data type. In this case, multiple drivers are permitted on the net as in Verilog-2001.
- A variable data type is not permitted on either side of an **`inout`** port.
- A **ref** port shall be connected to an equivalent variable data type. References to the port variable are treated as hierarchal references to the variable it is connected to in its instantiation. This kind of port may not be left unconnected

l12.8.2 Port Connection Rules for nets

If a port declaration has a *wire* type (which is the default), or any other net type, then its direction controls how it can be connected as follows:

- An **`input`** may be connected to any expression of a compatible data type. If left unconnected, it has the value `'z`.
- An **`output`** may be connected to a net type (or a concatenation of net types) or a compatible variable type (or a concatenation of variable types).
- An **`inout`** may be connected to a net type (or a concatenation of net types) or left unconnected, but not to a variable type.

Note that where the data types differ between the port declaration and connection, an initial value change event may be caused at time zero.

12.8.3 Port Connection Rules for interfaces

A port declaration may be a generic **`interface`** or named interface type.. An interface port instance must always be connected to an interface instance or a higher-level interface port. An interface port cannot be left unconnected.

If a port declaration has a generic **`interface`** type, then it can be connected to an interface instance of any type.If a port declaration has a named interface type, then it must be connected to an interface instance of the identical type.

See Section XX for more port connection rules with interfaces.

12.8.4 Compatible Port Types

The same rules for assignment compatibility are used for compatible port types for ports declared as an **`input`** or an **`output`** variable, or for **`output`** ports connected to variables. SystemVerilog does not change any of the other port connection compatibility rules

12.8.5 Unpacked array ports and arrays of instances

For an unpacked array port, the port and the array connected to the port must have the same number of unpacked dimensions, and each dimension of the port must have the same size as the corresponding dimension of the array being connected.
If the size and type of the port connection match the size and type of a single instance port, the connection shall be made to each instance in an array of instances.
If the port connection is an unpacked array, the unpacked array dimensions of each port connection shall be compared with the dimensions of the instance array. If they match exactly in size, each element of the port connection shall be matched to the port left index to left index, right index to right index. If they do not match it shall be considered an error.
If the port connection is a packed array, each instance shall get a part-select of the port connection, starting with all right-hand indices to match the right most part-select, and iterating through the right most dimension first. Too many or too few bits to connect all the instances shall be considered an error.