ERRATA 319: Port connection rules

The two sections 19.11.3 and 11.11.4 are displayed below with the changes proposed for fixing this errata.

19.11.3 Instantiation using implicit .name port connections

SystemVerilog adds the capability to implicitly instantiate ports using a .name syntax if the instance-port name and equivalent type match the connecting ~~variable~~declaration-port name and type. This enhancement eliminates the requirement to list a port name twice when the port name and ~~signal~~ declaration name are the same, while still listing all of the ports of the instantiated module for documentation purposes.

In the following alu_accum3 example, all of the ports of the instantiated alu module match the names of the ~~variables~~ declarations connected to the ports, except for the unconnected zero port, which is listed using a named port connection, showing that the port is unconnected. Implicit .name port connections are made for all name and equivalent type matching connections on the instantiated module.

In the same alu_accum3 example, the accum module has an 8-bit port called dataout that is connected to a 16-bit bus called dataout. Because the internal and external sizes of dataout do not match, the port must be connected by name, showing which bits of the 16-bit bus are connected to the 8-bit port. The datain port on the accum is connected to a bus by a different name (alu_out), so this port is also connected by name. The clk and rst_n ports are connected using implicit .name port connections. Also in the same alu_accum3 example, the xtend module has an 8-bit output port called dout and a 1- bit input port called din. Since neither of these port names match the names (or sizes) of the connecting ~~variables~~declarations, both are connected by name. The clk and rst_n ports are connected using implicit .name port connections.

```
module alu_accum3 (
output [15:0] dataout,
input [7:0] ain, bin,
input [2:0] opcode,
input clk, rst_n);
wire [7:0] alu_out;
alu alu (.alu_out, .zero(), .ain, .bin, .opcode);
accum accum (.dataout(dataout[7:0]), .datain(alu_out), .clk, .rst_n);
xtend xtend (.dout(dataout[15:8]), .din(alu_out[7]), .clk, .rst_n);
endmodule
```

A *.port_identifier* port connection is semantically equivalent to the named port connection *.port_identifier*(~~name~~port_identifier) ~~port connection~~ with the following exceptions:

— The port connection shall not create an implicit wire declaration.

— The declarations on each side of the port connection shall have equivalent data types.

— An implicit *.port_identifier* port connection between nets of two dissimilar net types shall generate an error when it is a warning in an explicit named port connection as required by the Verilog standard.

~~— The identifier referenced by *.port_identifier* shall not create an implicit wire declaration.~~

~~— It shall be illegal for a *.port_identifier* port connection to create an implicit cast. This includes truncation or padding.~~

~~— A conversion between a 2-state and 4-state type of the same bit length is a legitimate cast.~~

~~— It shall be an error if a *.port_identifier* port connection between two dissimilar net types would generate a~~

~~warning message as required by the Verilog standard.~~

19.11.4 Instantiation using implicit .* port connections

SystemVerilog adds the capability to implicitly instantiate ports using a .* syntax for all ports where the instance-port name ~~and type~~ matches the connecting ~~variable~~ port name and their data types are equivalent-~~type~~. This enhancement eliminates the requirement to list any port where the name and type of the connecting ~~variable~~ declaration match the name and equivalent type of the instance port. This implicit port connection style is used to indicate that all port names and types match the connections where emphasis is placed only on the exception ports. The implicit .* port connection syntax can greatly facilitate rapid block-level testbench generation where all of the testbench ~~variables~~ declarations are chosen to match the instantiated module port names and types.

In the following alu_accum4 example, all of the ports of the instantiated alu module match the names of the ~~variables~~ declarations connected to the ports, except for the unconnected zero port, which is listed using a named port connection, showing that the port is unconnected. The implicit .* port connection syntax connects all other ports on the instantiated module.

In the same alu_accum4 example, the accum module has an 8-bit port called dataout that is connected to a 16-bit bus called dataout. Because the internal and external sizes of dataout do not match, the port must be connected by name, showing which bits of the 16-bit bus are connected to the 8-bit port. The datain port on the accum is connected to a bus by a different name (alu_out), so this port is also connected by name. The clk and rst_n ports are connected using implicit .* port connections. Also in the same alu_accum4 example, the xtend module has an 8-bit output port called dout and a 1- bit input port called din. Since neither of these port names match the names (or sizes) of the connecting ~~variables~~ declarations, both are connected by name. The clk and rst_n ports are connected using implicit .* port connections.

```
module alu_accum4 (
output [15:0] dataout,
input [7:0] ain, bin,
input [2:0] opcode,
input clk, rst_n);
wire [7:0] alu_out;
alu alu (.*, .zero());
accum accum (.*, .dataout(dataout[7:0]), .datain(alu_out));
xtend xtend (.*, .dout(dataout[15:8]), .din(alu_out[7]));
endmodule
```

An implicit .* port connection is semantically equivalent to a default .name port connection for every port declared in the instantiated module.

A named port connection can be mixed with a .* connection to override the
port connection to a different expression or to leave the port unconnected.

When the implicit .* port connection is mixed in the same instantiation with named port connections, the implicit .* port connection token can be placed anywhere in the port list. The .* token can only appear at most once in the port list.

Modules can be instantiated into the same parent module using any combination of legal positional, named, implicit .name connected and implicit .* connected instances as shown in alu_accum5 example.

```
module alu_accum5 (
output [15:0] dataout,
```



```
input [7:0] ain, bin,
input [2:0] opcode,
input clk, rst_n);
wire [7:0] alu_out;
// mixture of named port connections and
// implicit .name port connections
alu alu (.ain(ain), .bin(bin), .alu_out, .zero(), .opcode);
// positional port connections
accum accum (dataout[7:0], alu_out, clk, rst_n);
// mixture of named port connections and implicit .* port connections
xtend xtend (.dout(dataout[15:8]), .*, .din(alu_out[7]));

endmodule
```

