12.7 Module instances

[BNF excerpt to be inserted after BNF is approved]

```
BNF section A.4.1.1 - Module instantiation

module_instance ::= name_of_instance ( [ list_of_port_connections ] )

name_of_instance ::= module_instance_identifier { range }

list_of_port_connections ::=
          ordered_port_connection { , ordered_port_connection }
        | dot_named_port_connection { , dot_named_port_connection }
        | { named_port_connection , } dot_star_port_connection
          { , named_port_connection }

ordered_port_connection ::= { attribute_instance } [ expression ]

named_port_connection ::= { attribute_instance } .port_identifier ( [ expression ] )

dot_named_port_connection ::=
          { attribute_instance } .port_identifier
        | named_port_connection

dot_star_port_connection ::= { attribute_instance } .*
```

CLIFF-NOTE: named port instantiation is now covered by

```
| dot_named_port_connection { , dot_named_port_connection }
```

because a dot_named_port_connection is now

```
dot_named_port_connection ::=
          { attribute_instance } .port_identifier
        | named_port_connection
```

END-CLIFF-NOTE

Syntax 12-20—Module instance syntax

A module can be used (instantiated) in two ways, hierarchical or top level. Hierarchical instantiation allows more than one instance of the same type. The module name can be a module previously declared or one declared later. Actual parameters can be named or ordered. ~~and port~~ Port connections can be ~~named~~ named, ~~or~~ ordered or implicitly connected. They can be nets, variables, or other kinds of interfaces, events, or expressions. See below for the connection rules.

Editor's note: Open issue from meeting 20: Cliff to send wording for .* and .name port connections.

Consider an alu-accumulator (alu_accum) example module that includes instantiations of an alu module, an accumulator register (accum) module and a sign-extension (xtend) module. The module headers for the three instantiated modules are shown in the following example code:

CLIFF-NOTE:

I hate all of the "output reg" declarations in these modules. Did we fix this in SystemVerilog 3.0 or was I blown off on this enhancement request and am I going to have to revisit this in 3.1?

We probably should look at all of the new examples and make sure that all of the output regs are declared if this did not pass.

END-CLIFF-NOTE

```
module alu (
  output reg [7:0] alu_out,
  output reg       zero,
  input      [7:0] ain, bin,
  input      [2:0] opcode);

  // RTL code for the alu module
endmodule

module accum (
  output reg [7:0] dataout,
  input      [7:0] datain,
  input            clk, rst_n);

  // RTL code for the accumulator module
endmodule

module xtend (
  output reg [7:0] dout,
  input            din,
  input            clk, rst_n);

  // RTL code for the sign-extension module
endmodule
```

12.7.1 Instantiation using positional port connections

Verilog has always permitted instantiation of modules using positional port connections as shown in the alu_accum1 module example.

```
module alu_accum1 (
  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, , ain, bin, opcode);

  accum accum (dataout[7:0], alu_out, clk, rst_n);

  xtend xtend (dataout[15:8], alu_out[7], clk, rst_n);

endmodule
```

As long as the connecting variables are order correctly and are the same size as the instance-ports that they are connected to, there will be no warnings and the simulation will work as expected.

12.7.2 Instantiation using named port connections

Verilog has always permitted instantiation of modules using named port connections as shown in the alu_accum2 module example.

```
module alu_accum2 (
  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(alu_out), .zero(),
             .ain(ain), .bin(bin), .opcode(opcode));

  accum accum (.dataout(dataout[7:0]), .datain(alu_out),
             .clk(clk), .rst_n(rst_n));

  xtend xtend (.dout(dataout[15:8]), .din(alu_out[7]),
             .clk(clk), .rst_n(rst_n));

endmodule
```

Named port connections do not have to be ordered the same as the ports of the instantiated module. The variables connected to the instance ports must be the same size or a port-size mismatch warning will be reported.

12.7.3 Instantiation using implicit .name port connections

SystemVerilog adds the capability to implicitly instantiate ports using a .name syntax if the instance-port name and size match the connecting variable-port name and size. This enhancement eliminates the requirement to list a port name twice when the port name and signal 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 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 size 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 data out. 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, 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
```

Implicit .name port connections do not have to be ordered the same as the ports of the instantiated module.

The following rules apply to implicit .name port connections:

- For an implicit .name port connection to be legal, the connecting variable name must match the port name of the instantiated module.
- For an implicit .name port connection to be legal, the connecting variable size must match the port size of the instantiated module.
- If implicit .name port connections are used in an instantiation, named port connections must be used if the connecting variable name or size do not match the name or size of the instantiated port.
- Implicit .name port connections cannot be used in the same instantiation with positional port connections.
- Implicit .name port connections may be used in the same instantiation with named port connections.
- Implicit .name port connections cannot be used in the same instantiation with implicit .* port connections.
- If implicit .name port connections are used in an instantiation, all unconnected ports must be shown using named port connections.
- The order of the implicit .name port connections does not have to match the port-order of the instantiated module.
- All connecting variables must be explicitly declared, either as a port in the parent module (following the rules of Verilog-2001) or as an explicit net or variable of one or more bits.

12.7.4 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 size match the connecting variable-port name and size. This enhancement eliminates the requirement to list any port where the name and size of the connecting variable match the name and size of the instance port. This implicit port connection style is used to indicate that all port names and sizes 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 are chosen to match the instantiated module port names and sizes.

In the following alu_accum4 example, all of the ports of the instantiated alu module match the names of the variables 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 data out. 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, 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
```

The following rules apply to implicit .* port connections:

- For an implicit .* port connection to be legal, all implicitly connected ports must have a connecting variable name to match the port name of the instantiated module.
- For an implicit .* port connection to be legal, all implicitly connected ports must have a connecting variable size to match the port size of the instantiated module.
- If implicit .* port connections are used in an instantiation, named port connections must be used wherever connecting variable names or sizes do not match the names or sizes of instantiated ports.
- Implicit .* port connections cannot be used in the same instantiation with positional port connections.
- Implicit .* port connections may be used in the same instantiation with named port connections.
- Implicit .* port connections cannot be used in the same instantiation with implicit .name port connections.
- If implicit .* port connections are used in an instantiation, all unconnected ports must be shown using named port connections.
- 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.
- All connecting variables must be explicitly declared, either as a port in the parent module (following the rules of Verilog-2001) or as an explicit net or variable of one or more bits.

Modules may 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
```