Add after 19.4.3 **An example of connecting a port bundle to a generic interface**

19.4.3+ Modport expressions

A modport expression allows elements of arrays and structures, concatenations of elements, aggregate expressions of elements declared in an interface to be included in a modport list. This modport expression is explicitly named with a port identifier, visible only through the modport connection.

Like explicitly named ports in a module port declaration, port identifiers exist in their own namespace for each modport list. When modport item is just a simple port identifier, that identifier is used as both a reference to an interface item and a port identifier. Once a port identifier has been defined, there shall not be another port definition with this same name.

For example:

```
interfce I;
   logic [7:0] r;
   const int x=1;
   bit R;
    modport A (output .P(r[3:0]), input .Q(x), R);
    modport B (output .P(r[7:4]), input .Q(2), R);
endinterface

module M ( interface i);,
       initial i.P = i.Q;
endmodule

module top;
   I i1;
   M u1 (i1.A);
   M u2 (i1.B);
   initial #1 $display("%b", i1.r); // displays 00010010
endmodule
```

The self-determined type of the port expression becomes the type for the port. If the port expression is to be an aggregate expression, then a cast must be used since self-determined aggregate expressions are not allowed. The port_expression must resolve to a legal expression for type of module port (See section 18.8 – Port Connection Rules). In the example above, the Q port could not be an output or inout because the port expression is a constant. The port expression is optional because ports can be defined that do not connect to anything internal to the port

Add after **18.5 Port declarations**

18.5+ List of Port expressions

Verilog 1364-2001 created a *list_of_port_declarations* alternate style which minimized the duplication of data used to specify the ports of a module. SystemVerilog adds add an explicitly named port declaration to that style, allowing elements of arrays and structures, concatenations of elements, or aggregate expressions of elements declared in a module, interface or program to be specified on the port list.

Like explicitly named ports in a module port declaration, port identifiers exist in their own namespace for each port list. When port item is just a simple port identifier, that identifier is used as both a reference to an interface item and a port identifier. Once a port identifier has been defined, there shall not be another port definition with this same name.

For example:

```
module mymod(
   output .P1(r[3:0]),
   output .P2(r[7:4]),
   ref .Y(x),
   input bit R);

   logic [7:0] r;
   int x;
…
endmodule
```

The self-determined type of the port expression becomes the type for the port. If the port expression is to be an aggregate expression, then a cast must be used since self-determined aggregate expressions are not allowed. The port_expression must resolve to a legal expression for type of module port (See section 18.8 – Port Connection Rules). The port expression is optional because ports can be defined that do not connect to anything internal to the port

BNF change

In section A.2.3, replace

list_of_port_identifiers ::= port_identifier { unpacked_dimension }
{ , port_identifier { unpacked_dimension } }
list_of_modport_port_identifiers ::= port_identifier { , port_identifier }

with

list_of_port_identifiers ::= port_identifier { unpacked_dimension }
{ , port_identifier { unpacked_dimension } }
| . port_identifier ([expression]) { . port_identifier ([expression]) }
list_of_modport_port_identifiers ::= port_identifier { , port_identifier }
| . port_identifier ([expression]) { . port_identifier ([expression]) }