Insert after 18.7.4

18.7.5 Instantiation using function call syntax

SystemVerilog also allows a module or primitive to be instantiated using function call syntax in wire declarations or continuous assignments. The first port must be an output, and this is used to return the value to the call.

```
function_call ::= hierarchical_function_identifier { attribute_instance } [ ( list_of_arguments ) ]
          | instantiation_call

instantiation_call ::= n_input_gatetype [delay2] ( input_terminal {, input_terminal})
          | module_identifier [parameter_value_assignment] ([list_of_port_connections] )
```

This feature allows netlists to be written more concisely. For example:

```
wire w = a ? and(b,c) : or(d, xor(e,f)); // 1 mux and 3 gates
```

The instance name must be synthesized from the left hand side name if it is an assignment. Where the function call is embedded in another expression, like the **xor** in the example, a static variable of the appropriate data type must be created to hold the output value. It should also be given a synthesized name.

Parameters can be passed as well. For example:

```
module adderchain #(parameter n=1)( output [15:0] sum, input a[n]);
wire [15:0] s [0:n-1];
generate for ( genvar I = 0; I < a.size; I++) begin: loop
    if (I > 0) assign s[I] = s[I-1] + a[I];
    else assign s[0] = a[1] + a[0];
end
endgenerate
endmodule

wire [15:0] S = adderchain #(4)(a,b,c,d);
```

Note that this instantiation syntax does not apply to interfaces or programs, because they do not represent hardware.

BNF changes to A.8.2

Replace
function_call ::= hierarchical_function_identifier { attribute_instance } [(list_of_arguments)]

With
function_call ::= hierarchical_function_identifier { attribute_instance } [(list_of_arguments)]
          | instantiation_call

instantiation_call ::= n_input_gatetype [delay2] (input_terminal {, input_terminal})
          | module_identifier [parameter_value_assignment] ([list_of_port_connections])