The proposal is aligned with P1800-2008 Draft 4

Elaboration-time user error messages

Objectives:

- To enable user controlled verification of parameter values during model elaboration and issuing of informative message, the proposal introduces new elaboration action system tasks.

21 Compiler directives

Add new clause

21.14 Elaboration-time messages

It is often necessary to validate the actual parameter values used in a SystemVerilog model and report any error without generating the executable simulation model. This is achieved by using elaboration action system tasks. The tasks shall be called outside procedural code and their activation may be controlled by conditional generate constructs. It is an error if such a task is called from within a procedure.

```
elaboration_action ::=
        $elab_fatal(list_of_arguments);
      | $elab_error(list_of_arguments);
      | $elab_warning(list_of_arguments);
      | $elab_info(list_of_arguments);
```

Syntax 21-?? Elaboration action syntax [Note to the Editor: Please assign appropriate number]

`list_of_arguments` may only contain a formatting string and constant expressions, including constant function calls. If a call to such a task remains in the elaborated model after any generate block expansion, the task is executed. Depending on the severity of the task the elaboration may be aborted or continue to successful completion. If there are more than one elaboration action task call present then they are executed in any order.

If `$elab_fatal` is executed then after outputting the message the run-time code generation is immediately aborted. This also means that any remaining elaboration action task calls may not be executed.

If `$elab_error` is executed then the message is issued and the elaboration continues. However, no simulation run-time code is generated.

The other two tasks `$elab_warning` and `$elab_info` only output their text message but do not affect the rest of the elaboration and the simulation.

All of the elaboration action system system tasks shall print a tool-specific message, indicating the severity of the exception condition and specific information about the condition, which shall include the following information:
— The file name and line number of the severity system task call. The file name and line number shall be same as `__LINE__ and ` __FILE__ compiler directives respectively.

— The hierarchical name of the scope in which the severity system task call is made.

The tool-specific message shall include the user-defined message if specified.

Example: Sometimes it is desirable to validate elaboration-time constants, such as bounds on a parameter, in a way that can be enforced during model elaboration. In this example, if the module parameter value is outside the range 1 to 8, an error is issued and the model elaboration is aborted.

```
module test #(N=12) (input [N-1:0] in, output [N-1:0] out);
   if ((N < 0) && (N > 8)) // conditional generate construct
      $elab_error("Parameter N  has an invalid value of %0d", N);
   assign out = in;
endmodule
```

Example: In this simple example, the generate construct builds a concatenation (##1) of subsequences, each of length 1, over a bit from a vector passed as argument to the top sequence definition. Elaboration messages are used to indicate if the vector is just a scalar, otherwise it will issue an information message indicating which conditional branch was taken.

```
   generate
       if ($bits(vect) == 1) begin : err $elab_error("Not a vector"); end
       for (genvar i = 0; i < $bits(vect); i++) begin : Loop
              if (i==0) begin : Cond
                    sequence t; vect[0]; endsequence
                    $elab_info("i=0 branch taken");
             end : Cond
             else begin : Cond
                    sequence t; vect[i] ##1 Loop[i-1].Cond.t; endsequence
                    $elab_info("i != 0 branch taken")
             end : Cond
       end : Loop
   endgenerate
   // instantiate the last generated sequence in a property
   property p;
       @(posedge clk) trig |-> Loop[$bits(vect)-1].Cond.t;
   endproperty
```

In A.1.4 Module items

ADD right after the title

elaboration_action ::=
$elab_fatal(list_of_arguments);
| $elab_error(list_of_arguments);
| $elab_warning(list_of_arguments);
| $elab_info(list_of_arguments);

REPLACE

module_common_item ::=
module_or_generate_item_declaration
| interface_instantiation
| program_instantiation
| concurrent_assertion_item

| bind_directive
| continuous_assign
| net_alias
| initial_construct
| final_construct
| always_construct
| loop_generate_construct
| conditional_generate_construct

WITH

module_common_item ::=
module_or_generate_item_declaration
| interface_instantiation
| program_instantiation
| concurrent_assertion_item
| bind_directive
| continuous_assign
| net_alias
| initial_construct
| final_construct
| always_construct
| loop_generate_construct
| conditional_generate_construct
| elaboration_action

In A.1.7 Program items

REPLACE

program_generate_item[36] ::=
loop_generate_construct
| conditional_generate_construct
| generate_region

WITH

program_generate_item[36] ::=
loop_generate_construct
| conditional_generate_construct
| generate_region
| elaboration_action