Annex A

Syntax

This annex contains the formal syntax definition of Verilog-AMS HDL. The conventions used are described in Section 1. The Verilog-AMS HDL syntax extends and modifies the *IEEE 1364-2001 Verilog HDL* syntax. This syntax should be read in conjunction with the *IEEE 1364-2001 Verilog HDL* syntax. The following type of syntax categories are present in this annex:

1. *IEEE 1364-2001 Verilog HDL* syntax categories are represented in blue with cyan keywords and punctuation. This syntax is present for readability.
2. Verilog-AMS HDL syntax categories are represented in black with red keywords and punctuation. These syntax categories are extensions to the *IEEE 1364-2001 Verilog HDL* syntax.
3. Modified *IEEE 1364-2001 Verilog HDL* syntax categories are displayed blue with cyan keywords and punctuation with the Verilog-AMS extensions represented in black with red keywords and punctuation. The categories in this annex supersedes the same categories in the *IEEE 1364-2001 Verilog HDL* annex A.

A.1 Source text

A.1.1 Library source text

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.1.2 Configuration source text

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.1.3 Module and primitive source text

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

```
description ::=
    module_declaration
    | udp_declaration
    | discipline_declaration
```

```
    | nature_declaration
    | connectrules_specification
module_keyword ::= module | macromodule | connectmodule
```

NOTE: Semantically restrict connectmodule port list to be ordered only
NOTE: Semantically restrict connectmodule port list to be size 2 only
NOTE: Semantically restrict connectmodule ports to be scalar.
NOTE: Semantically restrict a connectmodule ports to be non-empty

A.1.4 Module parameters and ports

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.1.5 Module Items

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

```
module_item ::=
     module_or_generate_item
    | port_declaration ;
    | { attribute_instance } generated_instantiation
    | { attribute_instance } local_parameter_declaration
    | { attribute_instance } parameter_declaration
    | { attribute_instance } specify_block
    | { attribute_instance } specparam_declaration
    | { attribute_instance } analog_module_item
non_port_module_item ::=
     module_or_generate_item
    | port_declaration ;
    | { attribute_instance } generated_instantiation
    | { attribute_instance } local_parameter_declaration
    | { attribute_instance } module_or_generate_item
    | { attribute_instance } parameter_declaration
    | { attribute_instance } specify_block
    | { attribute_instance } specparam_declaration
    | { attribute_instance } analog_module_item
analog_module_item ::=
    analog_module_item_declaration
    | analog_block
analog_module_item_declaration ::=
    ground_declaration
    | branch_declaration
    | analog_function_declaration
```

NOTE: removed "module_items" syntax item and added the analog_module_item and analog_module_item_declaration..
NOTE: Semantically restrict analog block to appear zero or one time in a module declaration.
NOTE: module items cannot appear within a generate item.
NOTE: removed duplicate "parameter_override" syntax.
NOTE: AMS declarations and analog blocks now can be prefixed with attributes.

A.1.6 Natures

nature_declaration ::=
nature nature_identifier [**:** parent_nature] **;**
{ nature_item }
endnature
parent_nature ::=
nature_identifier
| discipline_identifier **.** potential_or_flow
nature_item ::= nature_attribute
nature_attribute ::= nature_attribute_identifier **=** nature_attribute_expression **;**
nature_attribute_expression ::=
constant_expression
| nature_identifier
| nature_access_identifier

NOTE: ';' added to nature declaration. Update Annex D and backward compatibility sections.
NOTE: nature attribute identifiers move to section A.11.3
NOTE: nature attribute expression syntax item defined since constant expression cannot handle nature identifiers and nature access identifiers.

A.1.7 Disciplines

discipline_declaration ::=
discipline discipline_identifier **;**
{ discipline_item }
enddiscipline
discipline_item ::=
nature_binding
| discipline_domain_binding
| nature_attribute_override
nature_binding ::= potential_or_flow nature_identifier **;**
potential_or_flow ::= **potential** | **flow**
discipline_domain_binding ::= **domain** discrete_or_continuous **;**
discrete_or_continuous ::= **discrete** | **continuous**
nature_attribute_override ::= potential_or_flow **.** nature_attribute

NOTE: ';' added to discipline declaration. Update Annex D and backward compatibility sections.

A.1.8 Connectrule Specifications

connectrules_specification ::=
connectrules connectrules_identifier **;**
{ connectrules_item }
endconnectrules
connectrules_item ::=
connect_insertion
| connect_resolution
connect_insertion ::=
connect connectmodule_identifier [connect_mode] [parameter_value_assignment] [connect_port_overrides] **;**

connect_mode ::= **merged** | **split**
connect_port_overrides ::=
    discipline_identifier **,** discipline_identifier
    | **input** discipline_identifier **, output** discipline_identifier
    | **output** discipline_identifier **, input** discipline_identifier
    | **inout** discipline_identifier **, inout** discipline_identifier
connect_resolution ::= **connect** discipline_identifier { , discipline_identifier } **resolveto** discipline_identifier **;**

NOTE: connect_insertion syntax reusing digital parameter override syntax. Now can have order and named overrides.
NOTE: changed keyword "merge" to "merged".

A.2 Declarations

A.2.1 Declaration types

A.2.1.1 Module parameter declarations

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

parameter_declaration ::=
    parameter [**signed**] [range] list_of_param_assignments **;**
    | **parameter integer** list_of_param_assignments **;**
    | **parameter real** list_of_param_assignments **;**
    | **parameter realtime** list_of_param_assignments **;**
    | **parameter time** list_of_param_assignments **;**
    | **parameter** list_of_array_param_assignments **;**
    | **parameter real** list_of_array_param_assignments **;**
    | **parameter integer** list_of_array_param_assignments **;**

NOTE: Digital parameters now may use optional value_range syntax.
NOTE: Semantically restrict signed and first range for typeless analog and mixed parameter declarations.

A.2.1.2 Port declarations

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

inout_declaration ::= **inout** [discipline_identifier] [net_type] [**signed**] [range] list_of_port_identifiers
input_declaration ::= **input** [discipline_identifier] [net_type] [**signed**] [range] list_of_port_identifiers
output_declaration ::=
    output [discipline_identifier] [net_type] [**signed**] [range] list_of_port_identifiers
    | **output** [discipline_identifier] [**reg**] [**signed**] [range] list_of_port_identifiers
    | **output** [discipline_identifier] **reg** [**signed**] [range] list_of_variable_port_identifiers
    | **output** [output_variable_type] list_of_port_identifiers
    | **output** output_variable_type list_of_variable_port_identifiers

NOTE: Extend port direction declaration to specify disciplines. Semantically restrict port declarations that use ’net_type’, ’reg’ or ’signed’ to use a discrete disciplines only.

NOTE: Continuous ports cannot be tied to ground via the ground keyword or declaration.

A.2.1.3 Type declarations

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

```
net_declaration ::=
    net_type [ discipline_identifier ] [ signed ] [ delay3 ] list_of_net_identifiers ;
    | net_type [ discipline_identifier ] [ drive_strength ] [ signed ] [ delay3 ] list_of_net_decl_assignments ;
    | net_type [ discipline_identifier ] [ vectored | scalared ] [ signed ] [ delay3 ] list_of_net_identifiers ;
    | net_type [ discipline_identifier ] [ drive_strength ] [ vectored | scalared ] [ signed ] [ delay3 ]
        list_of_net_decl_assignments ;
    | trireg [ discipline_identifier ] [ charge_strength ] [ signed ] [ delay3 ] list_of_net_identifiers ;
    | trireg [ discipline_identifier ] [ drive_strength ] [ signed ] [ delay3 ] list_of_net_decl_assignments ;
    | trireg [ discipline_identifier ] [ charge_strength ] [ vectored | scalared ] [ signed ] range [ delay3 ]
        list_of_net_identifiers ;
    | trireg [ discipline_identifier ] [ drive_strength ] [ vectored | scalared ] [ signed ] range [ delay3 ]
        list_of_net_decl_assignments ;
    | discipline_identifier [ range ] list_of_net_identifiers ;
    | discipline_identifier [ range ] list_of_net_decl_assignments ;
    | wreal list_of_variable_identifiers ;
ground_declaration ::= ground [ discipline_identifier ] [ range ] list_of_net_identifiers ;
reg_declaration ::= reg [ discipline_identifier ] [ signed ] [ range ] list_of_variable_identifers ;
branch_declaration ::= branch ( branch_terminal [ , branch_terminal ] ) list_of_branch_identifiers ;
branch_terminal ::=
    port_identifier
    | port_identifer [ constant_expression ]
    | port_identifer [ constant_range_expression ]
    | net_identifier
    | net_identifer [ constant_expression ]
    | net_identifer [ constant_range_expression ]
```

NOTE: AMS 'integer_declaration' removed in favour of the equivalent 2001 syntax. Now, multiple dimension integer analog variables can be declared without initialization or scalar integer analog variables can be declared with initialization.
NOTE: AMS 'real_declaration' removed in favour of the equivalent 2001 syntax. Now, multiple dimension real analog variables can be declared without initialization or scalar real analog variables can be declared with initialization.
NOTE: Integer and real variables that are initialized but accessed in a module declaration are assumed to be digital variables.
NOTE: net discipline declarations have been merged in the 2001 net_declaration syntax.
NOTE: A ground net and its discipline now can be declared in a single declaration.
NOTE: Continuous net declarations may initialized the nets voltage only.
NOTE: Continuous nets now supports multiple dimensional arrays.
NOTE: Discrete nets may be associated with a discrete discipline either by a separate discipline declaration without initialization (backward compatible) or by specifying a discrete discipline in the net or reg declaration. However specifying a discrete discipline in both types of declaration for the same net is an error.
NOTE: Ground declaration can never be initialized.
NOTE: wreal nets now support multiple dimensions.
NOTE: Do we need to differentiate between analog and digital genvar variables? Currently using common syntax. Semantically restrict genvars to be used in either analog block or digital behavior, but not both.

NOTE: Updated branch syntax. Now supports part select of a continuous net for the branch terminals. Semantically restrict branch terminals to be continuous (implied but not explicit stated in section 3.9). This also is allowed "branch (x[10+:1]) brn;".
NOTE: Semantically restrict branch terminals in a branch declaration to be locally declared.

A.2.2 Declaration data types

A.2.2.4 Net and variable types

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.2.2.5 Strengths

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.2.2.6 Delays

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.2.3 Declaration lists

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

list_of_branch_identifiers ::= branch_identifier [range] { **,** branch_identifier [range] }
list_of_array_param_assignments ::= array_param_assignment { **,** array_param_assignment }

NOTE: Remove duplicate AMS 'list_of_genvar_identifiers' syntax in favour of the 2001 equivalent syntax.
NOTE: Removed old 1995 'list_of_identifiers' syntax item.

A.2.4 Declaration assignments

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

param_assignment ::= parameter_identifier = constant_expression { value_range }
array_param_assignment ::= parameter_identifier range = constant_concatenation { value_range }

NOTE: Added optional value range to digital, analog, and mixed parameter declarations.

A.2.5 Declaration ranges

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

value_range ::=
    from (value_range_expression **:** value_range_expression **)**
    | **from (** value_range_expression **:** value_range_expression **]**

```
    | from [ value_range_expression : value_range_expression )
    | from [ value_range_expression : value_range_expression ]
    | exclude ( value_range_expression : value_range_expression )
    | exclude ( value_range_expression : value_range_expression ]
    | exclude [ value_range_expression : value_range_expression )
    | exclude [ value_range_expression : value_range_expression ]
    | exclude constant_expression
value_range_expression ::= constant_expression | -inf | inf
```

NOTE: Removed multiple AMS 'range' and 'array_range' syntax items in favor of the 2001 'range' syntax item.

A.2.6 Function declarations

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.2.7 Analog function declarations

```
analog_function_declaration ::=
    analog function [ analog_function_type ] analog_function_identifier ;
    analog_function_item_declaration { analog_function_item_declaration }
    analog_statement
    endfunction
analog_function_type ::= integer | real
analog_function_item_declaration ::=
    analog_block_item_declaration
    | af_input_declaration
af_input_declaration ::= input [ range ] list_of_port_identifiers ;
```

NOTE: Update analog function declaration syntax to avoid conflicts with 2001 syntax categories.

A.2.8 Task declarations

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

```
tf_input_declaration ::=
    input [ discipline_identifier ] [ reg ] [ signed ] [ range ] list_of_port_identifiers
    | input [ task_port_type ] list_of_port_identifiers
tf_output_declaration ::=
    output [ discipline_identifier ] [ reg ] [ signed ] [ range ] list_of_port_identifiers
    | output [ task_port_type ] list_of_port_identifiers
tf_inout_declaration ::=
    inout [ discipline_identifier ] [ reg ] [ signed ] [ range ] list_of_port_identifiers
    | inout [ task_port_type ] list_of_port_identifiers
```

NOTE: Enable task port declarations to specify a discrete discipline or use 'default_discipline directive.

A.2.9 Block item declarations

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

block_reg_declaration ::= **reg** [**signed**] [discipline_identifier] [range] list_of_block_variable_identifiers **;**

NOTE: Enable block item register declarations to specify a discrete discipline or use ‘default_discipline directive.

A.2.10 Analog block item declarations

analog_block_item_declaration ::=
{ attribute_instance } parameter_declaration
| { attribute_instance } integer_declaration
| { attribute_instance } real_declaration

NOTE: Added attributes to analog block declarations in the same way as module scope declarations.

A.3 Primitive instances

A.3.1 Primitive instantiation and instances

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.3.2 Primitive strengths

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.3.3 Primitive terminals

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.3.4 Primitive gate and switch types

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.4 Module and generated instantiation

A.4.1 Module instantiation

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

NOTE: Reuse the 2001 module instantiation syntax instead of the AMS syntax. Now named/ordered port connection expressions are optional and named parameter override expressions are optional.

A.4.2 Generated instantiation

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.5 UDP declaration and instantiation

A.5.1 UDP declaration

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.5.2 UDP ports

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

```
udp_output_declaration ::=
      { attribute_instance } output port_identifier
      | { attribute_instance } output [ discipline_identifier ] reg port_identifier [ = constant_expression ]
udp_input_declaration ::= { attribute_instance } input list_of_port_identifiers
udp_reg_declaration ::= { attribute_instance } reg [ discipline_identifier ] variable_identifier
```

NOTE: Enable UDP ports and local registers to specify a discrete discipline or use ‘default_discipline directive.

A.5.3 UDP body

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.5.4 UDP instantiation

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.6 Behavioural statements

A.6.1 Continuous assignment statements

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.6.2 Procedural blocks and assignments

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.6.3 Parallel and sequential blocks

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.6.4 Statements

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.6.5 Timing control statements

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

*digital_event*_expression ::=
*digital*_expression
| *event*_identifier
| **posedge** *digital*_expression
| **negedge** *digital*_expression
| event_function
| d*igital*_event_expression **or** *digital*_event_expression

A.6.6 Conditional statements

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.6.7 Case statements

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.6.8 Looping statements

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.6.9 Task enable statements

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.7 Analog behavioral statements

A.7.1 Procedural blocks and assignments

analog_block ::=
    analog analog_statement
analog_procedural_assignment ::=
    lexpr **=** analog_expression **;**
lexpr ::=
     *integer*_identifier
    | *real*_identifier
    | array_element
array_element ::=
     *integer*_identifier **[** expression **]**
    | *real*_identifier **[** expression **]**
procedural_assignment ::=
    lexpr **=** expression **;**

A.7.2 Sequential blocks

analog_seq_block ::=
    begin [**:** *block*_identifier { block_item_declaration }]
     { analog_statement }
    end
seq_block ::=
    begin [**:** *block*_identifier { block_item_declaration }]
     { statement }
    end

A.7.3 Analog statements

analog_statement ::=
     analog_seq_block
    | analog_branch_contribution
    | analog_indirect_branch_assignment
    | analog_procedural_assignment
    | analog_conditional_statement
    | analog_for_statement
    | analog_case_statement
    | analog_event_controlled_statement
    | system_task_enable
    | statement
statement ::=
     seq_block
    | procedural_assignment
    | conditional_statement
    | loop_statement
    | case_statement

```
analog_statement_or_null ::=
      analog_statement | ;
statement_or_null ::=
      statement | ;
```

A.7.4 Contribution statements

```
analog_branch_contribution ::=
      bvalue <+ analog_expression ;
analog_indirect_branch_assignment ::=
      bvalue : nexpr == analog_expression ;
nexpr ::=
       bvalue
      | pvalue
      | ddt ( bvalue | pvalue )
      | idt ( bvalue | pvalue )
```

A.7.5 Genvar loop statement

```
analog_for_statement ::=
      for ( genvar_assignment ; genvar_expression ;
       genvar_assignment ) analog_statement
genvar_assignment ::=
      genvar_identifier = genvar_expression
```

A.7.6 Event control statements

```
event_control_statement ::=
      event_control statement_or_null
event_control ::=
       @ event_identifier
      | @ ( event_expression )
analog_event_expression ::=
        global_event
      | event_function
      | digital_expression
      | event_identifier
      | posedge digital_expression
      | negedge digital_expression
      | event_expression or event_expression
global_event ::=
       initial_step [ ( analysis_list ) ]
      | final_step [ ( analysis_list ) ]
analysis_list ::=
      analysis_name { , analysis_name }
analysis_name ::=
      " analysis_identifier "
event_function ::=
        cross_function
      | timer_function
```

cross_function ::=
    cross (arg_list **)**
timer_function ::=
    timer (arg_list **)**
driver_access_function ::=
    driver_update | **net_resolution**

A.7.7 Conditional statements

conditional_statement ::=
    if (expression **)** statement_or_null
    [**else** statement_or_null]
analog_conditional_statement ::=
    if (genvar_expression **)** analog_statement_or_null
    [**else** analog_statement_or_null]

A.7.8 Case statements

case_statement ::=
      case (expression **)** case_item {case_item} **endcase**
    | **casex (** expression **)** case_item {case_item} **endcase**
    | **casez (** expression **)** case_item {case_item} **endcase**
case_item ::=
      expression { **,** expression } **:** statement_or_null
    | **default** [**:**] statement_or_null
analog_case_statement ::=
      case (genvar_expression **)** analog_case_item { analog_case_item } **endcase**
    | **casex (** genvar_expression **)** analog_case_item { analog_case_item } **endcase**
    | **casez (** genvar_expression **)** analog_case_item { analog_case_item } **endcase**
analog_case_item ::=
      genvar_expression { **,** genvar_expression } **:** analog_statement_or_null
    | **default** [**:**] analog_statement_or_null

A.7.9 Analog loop statements

loop_statement ::=
      repeat (expression **)** statement
    | **while (** expression **)** statement
    | **for (** procedural_assignment **;** expression **;**
      procedural_assignment **)** statement

A.7.10 Analog task enable statements

system_task_enable ::=
    system_task_name [**(** expression { **,** expression } **)**] **;**
system_task_name ::=
    $identifier
Note: The **$** may not be followed by a space.

A.8 Specify section

A.8.1 Specify block declaration

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.8.2 Specify path declarations

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.8.3 Specify block terminals

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.8.4 Specify path delays

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.8.5 System timing checks

A.8.5.1 System timing check commands

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.8.5.2 System timing check cammand arguments

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.8.5.3 System timing check event definitions

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.9 Expressions

A.9.1 Concatenations

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.9.2 Function calls

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.9.3 Expressions

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.9.4 Primaries

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

*digital*_primary ::=
    primary

A.9.5 Expression left-side values

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.9.6 Operators

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.9.7 Numbers

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

*digital*_number ::=
    number

A.9.8 Strings

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

string ::=
    " { Any_ASCII_character_except_newline } "

A.10 AMS expressions

A.10.1 AMS concatentations

constant_array_expression ::=
    { constant_arrayinit_element { **,** constant_arrayinit_element } **}**
constant_arrayinit_element ::=
      constant_expression
    | *integer*_constant_expression **{** constant_expression **}**

constant_array_expression ::=
    { constant_array_init_element { , constant_array_init_element } }
constant_array_init_element ::=
      constant_expression
    | *integer*_constant_expression { , constant_expression }
concatenation ::=
    { expression { , expression } }

A.10.2 AMS function calls

function_call ::=
    *function*_identifier (expression { , expression })
system_function ::=
      $abstime | **$realtime** | **$temperature** | **$vt** | **$bound_step** | **$discontinuity**
    | **$driver_count** | **$driver_state** | **$driver_strength**

A.10.3 AMS expressions

analog_expression ::=
      expression
    | analog_operator (analog_operator_arg_list)
genvar_expression ::=
      genvar_primary
    | unary_operator genvar_primary
    | genvar_expression binary_operator genvar_primary
    | genvar_expression ? genvar_expression : genvar_expression
    | string
constant_expression ::=
      constant_primary
    | string
    | unary_operator constant_primary
    | constant_expression binary_operator constant_expression
    | constant_expression ? constant_expression : constant_expression
    | constant_array_expression
    | attribute_reference
    | built_in_function (const_arg_list)
const_arg_list ::=
    constant_expression { , constant_expression }
expression ::=
      primary
    | unary_operator primary
    | expression binary_operator expression
    | expression ? expression : expression
    | function_call
    | access_function_reference
    | built_in_function (arg_list)
    | system_function (arg_list)

A.10.4 Analog operators

analog_operator_arg_list ::=
  analog_operator_argument { **,** analog_operator_argument }
analog_operator_argument ::=
  | expression
  | constant_array_expression
  | analog_operator **(** analog_operator_arg_list **)**
analog_operator ::=
  ddt | **idt** | **idtmod** | **absdelay** | **transition** | **slew** |
  | **laplace_zd** | **laplace_zp** | **laplace_np** | **laplace_nd**
  | **zi_zp** | **zi_zd** | **zi_np** | **zi_nd** | **last_crossing** | **ac_stim**
  | limexp | **white_noise** | **flicker_noise** | **noise_table**

A.10.5 AMS primaries

genvar_primary ::=
    constant_primary
  | genvar_identifier
  | genvar_identifier [genvar_expression]
  | **analysis (** arg_list **)**
arg_list ::=
  argument { **,** argument}
argument ::=
    expression
  | constant_array_expression
constant_primary ::=
    number
  | *parameter*_identifier
  | *constant*_concatenation
primary ::=
    number
  | identifier
  | identifier **[** expression **]**
  | identifier **[** *digital_msb*_constant_expression **:** *digital_lsb*_constant_expression **]**
  | *digital*_concatenation
  | *digital*_multiple_concatenation
  | *digital*_function_call
  | **(***digital*_mintypmax_expression **)**
  | string
  | nexpr
  | **(** expression **)**

A.10.6 Nature attribute accessing

attribute_reference ::=
  *net*_identifier **.** pot_or_flow **.** attribute_identifier

A.10.7 Analog Probes

access_function_reference ::=
bvalue
| pvalue
bvalue ::=
access_identifier **(** analog_signal_list **)**
analog_signal_list ::=
*branch*_identifier
| *array_branch*_identifier [genvar_expression]
| net_or_port_scalar_expression
| net_or_port_scalar_expression **,** net_or_port_scalar_expression
net_or_port_scalar_expression ::=
*net_or_port*_identifier
| *array_net_or_port*_identifier [genvar_expression]
| *vector_net_or_port*_identifier [genvar_expression]
pvalue ::=
flow_access_identifier **(** < port_scalar_expression > **)**
port_scalar_expression ::=
*port*_identifier
| *array_port*_identifier [genvar_expression]
| *vector_port*_identifier [genvar_expression]

A.10.8 AMS operators

unary_operator ::=
+ | **-** | **!** | **~**
binary_operator ::=
+ | **-** | ***** | **/** | **%** | **==** | **===** | **!=** | **!==** | **&&** | **||**
| **<** | **<=** | **>** | **>=** | **&** | **|** | **^** | **^~** | **~^** | **>>** | **<<**

A.10.9 Mathematical functions

built_in_function ::=
ln | **log** | **exp** | **sqrt** | **min** | **max** | **abs** | **pow** | **ceil** | **floor**
| **sin** | **cos** | **tan** | **asin** | **acos** | **atan** | **atan2**
| **sinh** | **cosh** | **tanh** | **asinh** | **acosh** | **atanh** | **hypot**

A.10.10 AMS numbers

number ::=
decimal_number
| *digital*_octal_number
| *digital*_binary_number
| *digital*_hex_number
| real_number
decimal_number ::=
[sign] unsigned_num

```
real_number ::=
      [ sign ] unsigned_num . unsigned_num
    | [ sign ] unsigned_num [ . unsigned_num ] e [ sign ] unsigned_num
    | [ sign ] unsigned_num [ . unsigned_num ] E [ sign ] unsigned_num
    | [ sign ] unsigned_num [ . unsigned_num ] scale_factor
sign ::=
      +
    | -
unsigned_num ::=
    decimal_digit { _ | decimal_digit }
decimal_digit ::=
    0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
scale_factor ::=
    T | G | M | K | k | m | u | n | p | f | a
```

A.11 General

A.11.1 Attributes

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.11.2 Comments

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

```
comment ::=
      short_comment
    | long_comment
short_comment ::=
    // comment_text \n
long_comment ::=
    /* comment_text */
comment_text ::=
    { Any_ASCII_character }
```

A.11.3 Identifiers

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

```
analog_function_identifier ::= identifier
branch_identifier ::= identifier
connectmodule_identifier ::= module_identifier
connectrules_identifier ::= identifier
nature_identifier ::= identifier
nature_attribute_identifier ::=
      abstol
    | access
```

| **ddt_nature**
| **idt_nature**
| **units**
| identifier

discipline_identifier ::= identifier

identifier ::=
IDENTIFIER [{ **.** IDENTIFIER }]

NOTE: The period in identifier may not be preceded or followed by a space.

IDENTIFIER ::=
simple_identifier
| escaped_identifier

simple_identifier ::=
[**a-zA-Z_**]{ **a-zA-Z_$0-9** }

escaped_identifier ::=
\ { Any_ASCII_character_except_white_space } white_space

A.11.4 Identifier branches

Refer to the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

A.11.5 White space

The syntax below extends and modifies the syntax found in the equivalent section in *IEEE 1364-2001 Verilog HDL* annex A.

white_space ::=
space
| tab
| newline
| formfeeds