Annex A
Syntax

This annex contains the formal syntax definition of Verilog-AMS HDL. The conventions used are described in Section 1. Any category whose name begins with the italicized word *digital_* should be interpreted by its definition in the grammar given in *IEEE 1364-2001 Verilog HDL* Annex A, and not by the local definition given herein. When such a category is defined herein (e.g., *digital*_primary ::=), that definition should be taken to supersede the definition in *IEEE 1364-2001 Verilog HDL* when used for Verilog-AMS HDL. If any category is defined here and also in *IEEE 1364-2001 Verilog HDL*, it is a repetition of *IEEE 1364-2001 Verilog HDL*. This is done for the convenience of readers of this LRM.

The AMS annex A syntax has been reorganized into the 2001 annex A structure and color coding is used to identify any 2001 syntax. The syntax itself has not been modified. Duplicate 2001 syntax items are highlighted in blue. Updated or modified 2001 syntax items contain blue and black color coded syntax. Note, some 2001 syntax items have been used to define pure analog syntax (i.e. expression). Alternate AMS syntax item names are required.

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.

```
source_text ::=
      {description}
description ::=
      module_declaration
```

| digital_udp_declaration
| discipline_definition
| nature_definition
| connect_specification

module_declaration ::=
module_keyword *module*_identifier [*digital*_list_of_ports] **;**
[module_items]
endmodule

module_keyword ::=
module
| **macromodule**

connectmodule_declaration ::=
connectmodule module_identifier **(** connectmod_port **,** connectmod_port **) ;**
[module_items]
endmodule

connectmod_port ::=
*connectmod_port*_identifier

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_items ::=
{ module_item }
| analog_block

module_item ::=
module_item_declaration
| parameter_override
| module_instantiation
| *digital*_continuous_assignment
| *digital*_gate_instantiation
| *digital*_udp_instantiation
| *digital*_specify_block
| *digital*_initial_construct
| *digital*_always_construct

module_item_declaration ::=
parameter_declaration
| *digital*_input_declaration
| digital_output_declaration
| *digital*_inout_declaration
| ground_declaration
| integer_declaration
| real_declaration
| net_discipline_declaration
| genvar_declaration

| branch_declaration
| analog_function_declaration
| *digital*_function_declaration
| *digital*_net_declaration
| d*igital*_reg_declaration
| *digital*_time_declaration
| d*igital*_realtime_declaration
| *digital*_event_declaration
| *digital*_task_declaration

parameter_override ::=
defparam list_of_param_assignments **;**

A.1.6 Natures

nature_declaration ::=
nature nature_name
[nature_descriptions]
endnature

nature_name ::=
*nature*_identifier
| *nature*_identifier **:** parent_identifier

parent_identifier ::=
*nature*_identifier
| *discipline*_identifier**.flow**
| *discipline*_identifier**.potential**

nature_descriptions ::=
nature_description { nature_description }

nature_description ::=
attribute **=** constant_expression **;**

attribute ::=
abstol
| **access**
| **ddt_nature**
| **idt_nature**
| **units**
| *attribute*_identifier

A.1.7 Disciplines

discipline_declaration ::=
discipline *discipline*_identifier
[discipline_descriptions]
enddiscipline

discipline_descriptions ::=
discipline_description { discipline_description }

discipline_description ::=
nature_binding
| attr_override
| domain_binding

nature_binding ::=
    pot_or_flow *nature*_identifier **;**
attr_override ::=
    pot_or_flow **.** *attribute*_identifier **=** constant_expression **;**
pot_or_flow ::=
    potential
    | **flow**

domain_binding ::=
    domain discrete
    | **domain continuous**

A.1.8 Connectrule Specifications

connect_specification ::=
    connectrules *connectrule*_identifier **;**
    {connect_spec_item }
    endconnectrules
connect_spec_item ::=
    connect_insertion
    | connect_resolution
connect_insertion ::=
    connect *connect_module*_identifier connect_attributes
    [[direction] *discipline*_identifier **,** [direction] *discipline*_identifier] **;**
connect_attributes ::=
    [connect_mode] [**#(** attribute_list **)**]
connect_mode ::=
    merge
    | **split**
attribute_list ::=
    attribute
    | attribute_list **,** attribute
attribute ::=
    .*parameter*_identifier **(** expression **)**
direction ::=
    input
    | **output**
    | **inout**
discipline_list ::=
    *discipline*_identifier
    | discipline_list **,** *discipline*_identifier
connect_resolution ::=
    connect discipline_list **resolveto** *discipline*_identifier**;**

discipline_list ::=
    *discipline*_identifier
    | discipline_list **,** *discipline*_identifier

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 [opt_type] list_of_param_assignments **;**
opt_type ::=
    real
    | **integer**
declarator_init ::=
    *parameter*_identifier **=** constant_expression **{** opt_value_range **}**
    | *parameter_array*_identifier range **=** constant_param_arrayinit **{** opt_value_range **}**
opt_value_range ::=
    from value_range_specifier
    | **exclude** *value*_range_specifier
    | **exclude** *value*_constant_expression
value_range_specifier ::=
    start_range_spec expression1 **:** expression2 end_range_spec
value_range_specifier ::=
    start_paren expression1 **:** expression2 end_paren
start_paren ::=
    [| **(**
end_paren ::=
    ] | **)**
expression1 ::=
    constant_expression | **-inf**
expression2 ::=
    constant_expression | **inf**
constant_param_arrayinit ::=
    { param_arrayinit_element_list **}**
param_arrayinit_element_list
    param_arrayinit_element { **,** param_arrayinit_element }
param_arrayinit_element ::=
    constant_expression
    | **{** *replicator*_constant_expression **{**constant_expression**} }**

A.2.1.2 Port declarations

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

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.

integer_declaration ::=
  integer list_of_identifiers **;**
real_declaration ::=
  real list_of_identifiers **;**
var_name ::=
  *variable*_identifier
  | *array*_identifier array_range
ground_declaration ::=
  ground [range] list_of_nets ;
net_discipline_declaration ::=
  *discipline*_identifier [range] list_of_nets **;**
list_of_nets ::=
  *net*_identifier [range]
  | *net*_identifier [range] **,** list_of_nets
*digital*_net_declaration ::=
  *digital*_net_declaration
  | **wreal** [list_of_identifiers] **;**
branch_declaration ::=
  branch list_of_branches **;**
list_of_branches ::=
  terminals list_of_branch_identifiers
terminals ::=
  (net_or_port_scalar_expression **)**
  | **(** net_or_port_scalar_expression **,** net_or_port_scalar_expression **)**
genvar_declaration ::=
  genvar list_of_genvar_identifiers **;**

A.2.2 Delcaration 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_genvar_identifiers ::=
  *genvar*_identifier { **,** *genvar*_identifier }
list_of_branch_identifiers ::=
  *branch*_identifier [range]
  | *branch*_identifier [range] **,** list_of_branch_identifiers

list_of_identifiers ::=
    var_name { **,** var_name }
list_of_param_assignments ::=
    declarator_init { **,** declarator_init}

A.2.4 Declaration assignments

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

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.

array_range ::=
    [*upper_limit*_constant_expression **:** *lower_limit*_constant_expression **]**
range ::=
    [constant_expression **:** constant_expression **]**
range ::=
    [constant_expression **:** constant_expression **]**

A.2.6 Function declarations

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

A.2.7 Analog function declarations

analog_function_declaration ::=
    analog function [type] *function*_identifier **;**
    function_item_declaration { function_item_declaration }
    statement
    endfunction
type ::=
    integer
    | **real**
function_item_declaration ::=
    block_item_declaration
    | input_declaration

A.2.8 Task declarations

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

A.2.9 Block item declarations

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

A.2.10 Analog block item declarations

```
block_item_declaration ::=
      parameter_declaration
    | integer_declaration
    | real_declaration
```

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

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

```
module_instantiation ::=
    module_identifier [ parameter_value_assignment ] instance_list
instance_list ::=
    module_instance { , module_instance } ;
module_instance ::=
    name_of_instance  ( [ list_of_module_connections ] )
name_of_instance ::=
    module_instance_identifier [ range ]
```

```
list_of_module_connections ::=
      ordered_port_connection { , ordered_port_connection }
    | named_port_connection { , named_port_connection }
ordered_port_connection ::=
     net_expression
named_port_connection ::=
    . port_identifier ( net_expression )
parameter_value_assignment ::=
      # ( ordered_param_override_list )
    | # ( named_param_override_list )
ordered_param_override_list ::=
    constant_or_constant_array_expression { , constant_or_constant_array_expression }
named_param_override_list ::=
    named_param_override { , named_param_override }
named_param_override ::=
    . parameter_identifier ( constant_or_constant_array_expression )
constant_or_constant_array_expression ::=
      constant_expression
    | constant_array_expression
net_expression ::=
      net_identifier
    | net_identifier [ expression ]
    | net_identifier [ msb_constant_expression : lsb_constant_expression ]
    | net_concatenation
net_concatenation ::=
    { net_expression_list }
net_expression_list ::=
    net_expression { , net_expression }
```

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

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

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.

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