

# Appendix A : VHDL'93 AND VHDL'87 SYNTAX SUMMARY

---

`abstract_literal ::= decimal_literal | based_literal`

`access_type_definition ::= access subtype_indication`

`actual_designator ::=`  
    `expression`  
    `| signal_name`  
    `| variable_name`  
    `| file_name`  
    `| open`

`actual_parameter_part ::= parameter_association_list`

`actual_part ::=`  
    `actual_designator`  
    `| function_name ( actual_designator )`  
    `| type_mark ( actual_designator )`

`adding_operator ::= + | - | &`

`aggregate ::=`  
    `( element_association { , element_association } )`

`alias_declaration ::=`  
    `alias alias_designator [ : subtype_indication ] is`  
        `name [ signature ] ;`

`alias_designator ::= identifier | character_literal |`  
    `operator_symbol`

`allocator ::=`  
    `new subtype_indication`  
    `| new qualified_expression`

`architecture_body ::=`  
    `architecture identifier of entity_name is`  
        `architecture_declarative_part`  
    `begin`  
        `architecture_statement_part`  
    `end [ architecture ] [ architecture_simple_name ] ;`

`architecture_declarative_part ::=`  
    `{ block_declarative_item }`

`architecture_statement_part ::=`  
    `{ concurrent_statement }`

`array_type_definition ::=`  
    `unconstrained_array_definition |`  
    `constrained_array_definition`

`assertion ::=`  
    `assert condition`  
        `[ report_expression ]`  
        `[ severity_expression ]`

`assertion_statement ::= [ label : ] assertion ;`

`association_element ::=`  
    `[ formal_part => ] actual_part`

`association_list ::=`  
    `association_element { , association_element }`

`attribute_declaration ::=`  
    `attribute identifier : type_mark ;`

```

attribute_designator ::= attribute_simple_name
                        | shared_variable_declaration
                        | file_declaration
                        | alias_declaration
                        | component_declaration
                        | attribute_declaration
                        | attribute_specification
                        | configuration_specification
                        | disconnection_specification
                        | use_clause
                        | group_template_declaration
                        | group_declaration

attribute_name ::=
    prefix [ signature ] ' attribute_designator
    [ (expression) ]

attribute_specification ::=
    attribute attribute_designator of
    entity_specification is expression ;

base ::= integer

base_specifier ::= B | O | X

base_unit_declaration ::= identifier ;

based_integer ::=
    extended_digit { [ underline ] extended_digit }

based_literal ::=
    base # based_integer [ . based_integer ] #
    [ exponent ]

basic_character ::=
    basic_graphic_character | format_effector

basic_graphic_character ::=
    upper_case_letter | digit | special_character |
    space_character

basic_identifier ::=
    letter { [ underline ] letter_or_digit }

binding_indication ::=
    [ use entity_aspect ]
    [ generic_map_aspect ]
    [ port_map_aspect ]

bit_string_literal ::= base_specifier " bit_value "

bit_value ::= extended_digit { [ underline ]
    extended_digit }

block_configuration ::=
    for block_specification
        { use_clause }
        { configuration_item }
    end for ;

block_declarative_item ::=
    subprogram_declaration
    | subprogram_body
    | type_declaration
    | subtype_declaration
    | constant_declaration
    | signal_declaration
    | shared_variable_declaration
    | file_declaration
    | alias_declaration
    | component_declaration
    | attribute_declaration
    | attribute_specification
    | configuration_specification
    | disconnection_specification
    | use_clause
    | group_template_declaration
    | group_declaration

block_declarative_part ::=
    { block_declarative_item }

block_header ::=
    [ generic_clause
    [ generic_map_aspect ; ]
    [ port_clause
    [ port_map_aspect ; ] ]

block_specification ::=
    architecture_name
    | block_statement_label
    | generate_statement_label
    [ ( index_specification ) ]

block_statement ::=
    block_label :
        block [ ( guard_expression ) ] [ is ]
        block_header
        block_declarative_part
        begin
        block_statement_part
        end block [ block_label ] ;

block_statement_part ::=
    { concurrent_statement }

case_statement ::=
    [ case_label : ]
    case expression is
        case_statement_alternative
        { case_statement_alternative }
    end case [ case_label ] ;

case_statement_alternative ::=
    when choices =>
        sequence_of_statements

character_literal ::= ' graphic_character '

```

```

choice ::=
    simple_expression
    | discrete_range
    | element_simple_name
    | others

choices ::= choice { | choice }

component_configuration ::= -- VHDL'87
    for component_specification
        [ use binding_indication ; ]
        [ block_configuration ]
    end for ;

component_configuration ::= -- VHDL'93
    for component_specification
        [ binding_indication ; ]
        [ block_configuration ]
    end for ;

component_declaration ::=
    component_identifier [ is ]
        [ local_generic_clause ]
        [ local_port_clause ]
    end component [ component_simple_name ] ;

component_instantiation_statement ::= -- VHDL'87
    instantiation_label :
        component_name
        [ generic_map_aspect ]
        [ port_map_aspect ] ;

component_instantiation_statement ::= -- VHDL'93
    instantiation_label :
        instantiated_unit
        [ generic_map_aspect ]
        [ port_map_aspect ] ;

component_specification ::=
    instantiation_list : component_name

composite_type_definition ::=
    array_type_definition
    | record_type_definition

concurrent_assertion_statement ::=
    [ label : ] [ postponed ] assertion ;

concurrent_procedure_call_statement ::=
    [ label : ] [ postponed ] procedure_call ;

concurrent_signal_assignment_statement ::=
    [ label : ] [ postponed ]
        conditional_signal_assignment
    | [ label : ] [ postponed ]
        selected_signal_assignment

concurrent_statement ::=
    block_statement
    | process_statement
    | concurrent_procedure_call_statement
    | concurrent_assertion_statement
    | concurrent_signal_assignment_statement
    | component_instantiation_statement
    | generate_statement

condition ::= boolean_expression

condition_clause ::= until condition

conditional_signal_assignment ::=
    target <= options
        conditional_waveforms ;

conditional_waveforms ::=
    { waveform when condition else }
    waveform [ when condition ]

configuration_declaration ::=
    configuration_identifier of entity_name is
        configuration_declarative_part
        block_configuration
    end [ configuration ]
        [ configuration_simple_name ] ;

configuration_declarative_item ::=
    use_clause
    | attribute_specification
    | group_declaration

configuration_declarative_part ::=
    { configuration_declarative_item }

configuration_item ::=
    block_configuration
    | component_configuration

configuration_specification ::= -- VHDL'87
    for component_specification use
        binding_indication ;

configuration_specification ::= -- VHDL'93
    for component_specification binding_indication ;

constant_declaration ::=
    constant_identifier_list : subtype_indication
        [ := expression ] ;

constrained_array_definition ::=
    array_index_constraint of
        element_subtype_indication

```

```
constraint ::=
    range_constraint
    | index_constraint
```

```
context_clause ::= { context_item }
```

```
context_item ::=
    library_clause
    | use_clause
```

```
decimal_literal ::= integer [ . integer ] [ exponent ]
```

```
declaration ::=
    type_declaration
    | subtype_declaration
    | object_declaration
    | interface_declaration
    | alias_declaration
    | attribute_declaration
    | component_declaration
    | group_template_declaration
    | group_declaration
    | entity_declaration
    | configuration_declaration
    | subprogram_declaration
    | package_declaration
```

```
delay_mechanism ::= -- VHDL'93
    transport
    | [ reject time_expression ] inertial
```

```
design_file ::= design_unit { design_unit }
```

```
design_unit ::= context_clause library_unit
```

```
designator ::= identifier | operator_symbol
```

```
direction ::= to | downto
```

```
disconnection_specification ::=
    disconnect guarded_signal_specification after
    time_expression ;
```

```
discrete_range ::= discrete_subtype_indication | range
```

```
element_association ::=
    [ choices => ] expression
```

```
element_declaration ::=
    identifier_list : element_subtype_definition ;
```

```
element_subtype_definition ::= subtype_indication
```

```
entity_aspect ::=
    entity entity_name [ ( architecture_identifier ) ]
    | configuration configuration_name
    | open
```

```
entity_class ::=
    entity | architecture | configuration
    | procedure | function | package
    | type | subtype | constant
    | signal | variable | component
    | label | literal | units
    | group | file
```

```
entity_class_entry ::= entity_class [ <> ]
```

```
entity_class_entry_list ::=
    entity_class_entry { . entity_class_entry }
```

```
entity_declaration ::=
    entity identifier is
        entity_header
        entity_declarative_part
    [ begin
        entity_statement_part ]
    end [ entity ] [ entity_simple_name ] ;
```

```
entity_declarative_item ::=
    subprogram_declaration
    | subprogram_body
    | type_declaration
    | subtype_declaration
    | constant_declaration
    | signal_declaration
    | shared_variable_declaration
    | file_declaration
    | alias_declaration
    | attribute_declaration
    | attribute_specification
    | disconnection_specification
    | use_clause
    | group_template_declaration
    | group_declaration
```

```
entity_declarative_part ::=
    { entity_declarative_item }
```

```
-- VHDL'87
```

```
entity_designator ::= simple_name | operator_symbol
```

```
-- VHDL'93
```

```
entity_designator ::= entity_tag [ signature ]
```

```

entity_header ::=
    [ formal_generic_clause ]
    [ formal_port_clause ]

entity_name_list ::=
    -entity_designator { , entity_designator }
    | others
    | all

entity_specification ::=
    entity_name_list : entity_class

entity_statement ::=
    concurrent_assertion_statement
    | passive_concurrent_procedure_call_statement
    | passive_process_statement

entity_statement_part ::=
    { entity_statement }

-- VHDL'93
entity_tag :: simple_name | character_literal |
    operator_symbol

enumeration_literal ::= identifier | character_literal

enumeration_type_definition ::=
    ( enumeration_literal { , enumeration_literal } )

exit_statement ::=
    [ label : ] exit [ loop_label ] [ when condition ] ;

exponent ::= E [ + ] integer | E - integer

expression ::=
    relation { and relation }
    | relation { or relation }
    | relation { xor relation }
    | relation [ nand relation ]
    | relation [ nor relation ]
    | relation { xnor relation }

extended_digit ::= digit | letter

extended_identifier ::=
    \ graphic character { graphic character } \

factor ::=
    primary [ ** primary ]
    | abs primary
    | not primary

-- VHDL'87
file_declaration ::=
    file identifier : subtype_indication is [ mode ]
    file_logical_name ;

-- VHDL'93
file_declaration ::=
    file identifier_list : subtype_indication
    file_open_information ] ;

file_logical_name ::= string_expression

-- VHDL'93
file_open_information ::=
    [ open file_open_kind_expression ] is
    file_logical_name

file_type_definition ::=
    file of type_mark

floating_type_definition := range_constraint

formal_designator ::=
    generic_name
    | port_name
    | parameter_name

formal_parameter_list ::= parameter_interface_list

formal_part ::=
    formal_designator
    | function_name ( formal_designator )
    | type_mark ( formal_designator )

full_type_declaration ::=
    type identifier is type_definition ;

function_call ::=
    function_name [ ( actual_parameter_part ) ]

-- VHDL'87
generate_statement ::=
    generate_label : generation_scheme generate
    { concurrent_statement }
    end generate [ generate_label ] ;

-- VHDL'93
generate_statement ::=
    generate_label :
    generation_scheme generate
    [ { block_declarative_item }
    begin ]
    { concurrent_statement }
    end generate [ generate_label ] ;

```

```

generation_scheme ::=
    for generate_parameter_specification
    | if condition

generic_clause ::=
    generic ( generic_list ) ;

generic_list ::= generic_interface_list

generic_map_aspect ::=
    generic map ( generic_association_list )

graphic_character ::=
    basic_graphic_character | lower_case_letter |
    other_special_character

group_constituent ::= name | character_literal

group_constituent_list ::= group_constituent
    { , group_constituent }

group_declaration ::=
    group identifier : group_template_name
    ( group_constituent_list ) ;

group_template_declaration ::=
    group identifier is ( entity_class_entry_list ) ;

guarded_signal_specification ::=
    guarded_signal_list : type_mark

-- VHDL'87
identifier ::=
    letter { [ underline ] letter_or_digit }

-- VHDL'93
identifier ::=
    basic_identifier | extended_identifier

identifier_list ::= identifier { , identifier }

if_statement ::=
    [ if_label : ]

        if condition then
            sequence_of_statements
        { elsif condition then
            sequence_of_statements }
        [ else
            sequence_of_statements ]
        end if [ if_label ] ;

incomplete_type_declaration ::= type identifier ;

index_constraint ::= ( discrete_range
    { , discrete_range } )

index_specification ::=
    discrete_range
    | static_expression

index_subtype_definition ::= type_mark range <>

indexed_name ::= prefix ( expression { , expression } )

-- VHDL'93
instantiated_unit ::=
    [ component ] component_name
    | entity entity_name [ ( architecture_identifier ) ]
    | configuration configuration_name

instantiation_list ::=
    instantiation_label { , instantiation_label }
    | others
    | all

integer ::= digit { [ underline ] digit }

integer_type_definition ::= range_constraint

interface_constant_declaration ::=
    [ constant ] identifier_list : [ in ]
    subtype_indication [ := static_expression ]

interface_declaration ::=
    interface_constant_declaration
    | interface_signal_declaration
    | interface_variable_declaration
    | interface_file_declaration

interface_element ::= interface_declaration

-- VHDL'93
interface_file_declaration ::=
    file identifier_list : subtype_indication

interface_list ::=
    interface_element { ; interface_element }

interface_signal_declaration ::=
    [signal] identifier_list : [ mode ]
    subtype_indication [ bus ] [ := static_expression ]

interface_variable_declaration ::=
    [variable] identifier_list : [ mode ]
    subtype_indication [ := static_expression ]

iteration_scheme ::=
    while condition
    | for loop_parameter_specification

```

label ::= identifier	variable_declaration   file_declaration
letter ::= upper_case_letter   lower_case_letter	operator_symbol ::= string_literal
letter_or_digit ::= letter   digit	-- VHDL'87 options ::=
library_clause ::= library logical_name_list ;	[ guarded ] [ transport ]
library_unit ::=	-- VHDL'93 options ::= [ guarded ] [ delay_mechanism ]
primary_unit	
secondary_unit	
literal ::=	package_body ::=
numeric_literal	package body <i>package_simple_name</i> is
enumeration_literal	package_body_declarative_part
string_literal	end [ package body ] [ <i>package_simple_name</i> ] ;
bit_string_literal	
null	
logical_name ::= identifier	package_body_declarative_item ::=
logical_name_list ::= logical_name { , logical_name }	subprogram_declaration
logical_operator ::= and   or   nand   nor   xor   <u>xnor</u>	subprogram_body
loop_statement ::= [ /loop_label : ]	type_declaration
[ iteration_scheme ] loop	subtype_declaration
sequence_of_statements	constant_declaration
end loop [ /loop_label ] ;	<u>shared_variable_declaration</u>
	file_declaration
miscellaneous_operator ::= **   abs   not	alias_declaration
mode ::= in   out   inout   buffer   linkage	use_clause
multiplying_operator ::= *   /   mod   rem	<u>group_template_declaration</u>
name ::=	<u>group_declaration</u>
simple_name	
operator_symbol	package_body_declarative_part ::=
selected_name	{ package_body_declarative_item }
indexed_name	
slice_name	package_declaration ::=
attribute_name	package identifier is
	package_declarative_part
next_statement ::=	end [ package ] [ <i>package_simple_name</i> ] ;
[ label : ] next [ /loop_label ] [ when condition ] ;	
null_statement ::= [ label : ] null ;	package_declarative_item ::=
	subprogram_declaration
numeric_literal ::=	type_declaration
abstract_literal	subtype_declaration
physical_literal	constant_declaration
	signal_declaration
object_declaration ::=	<u>shared_variable_declaration</u>
constant_declaration	file_declaration
signal_declaration	alias_declaration
	component_declaration
	attribute_declaration
	attribute_specification
	disconnection_specification
	use_clause
	<u>group_template_declaration</u>
	<u>group_declaration</u>

```

package_declarative_part ::=
    { package_declarative_item }

parameter_specification ::=
    identifier in discrete_range

physical_literal ::= [ abstract_literal ] unit_name

physical_type_definition ::=
    range_constraint
    units
    base_unit_declaration
    { secondary_unit_declaration }
    end units [ physical_type_simple_name ]

port_clause ::=
    port ( port_list );

port_list ::= port_interface_list

port_map_aspect ::=
    port map ( port_association_list )

prefix ::=
    name
    | function_call

primary ::=
    name
    | literal
    | aggregate
    | function_call
    | qualified_expression
    | type_conversion
    | allocator
    | ( expression )

primary_unit ::=
    entity_declaration
    | configuration_declaration
    | package_declaration

procedure_call ::= procedure_name
    [ ( actual_parameter_part ) ]

procedure_call_statement ::=
    [ label : ] procedure_call ;

process_declarative_item ::=
    subprogram_declaration
    | subprogram_body
    | type_declaration
    | subtype_declaration
    | constant_declaration
    | variable_declaration
    | file_declaration
    | alias_declaration
    | attribute_declaration
    | attribute_specification
    | use_clause
    | group_template_declaration
    | group_declaration

process_declarative_part ::=
    { process_declarative_item }

process_statement ::=
    [ process_label : ]
    [ postponed ] process ( ( sensitivity_list ) ) [ is ]
    process_declarative_part
    begin
    process_statement_part
    end [ postponed ] process [ process_label ] ;

process_statement_part ::=
    { sequential_statement }

qualified_expression ::=
    type_mark ' ( expression )
    | type_mark ' aggregate

range ::=
    range_attribute_name
    | simple_expression direction simple_expression

range_constraint ::= range range

record_type_definition ::=
    record
    element_declaration
    { element_declaration }
    end record [ record_type_simple_name ]

relation ::=
    shift_expression [ relational_operator
    shift_expression ]

relational_operator ::=
    = | /= | < | <= | > | >=

return_statement ::=
    [ label : ] return [ expression ] ;

report_statement ::=
    [ label : ]
    report expression
    [ severity expression ] ;

```



```

scalar_type_definition ::=
    enumeration_type_definition |
    integer_type_definition |
    floating_type_definition |
    physical_type_definition

secondary_unit ::=
    architecture_body
    | package_body

secondary_unit_declaration ::=
    identifier = physical_literal ;

selected_name ::= prefix . suffix

selected_signal_assignment ::=
    with expression select
        target <= options selected_waveforms ;

selected_waveforms ::=
    { waveform when choices , }
    waveform when choices

sensitivity_clause ::= on sensitivity_list

sensitivity_list ::= signal_name { , signal_name }

sequence_of_statements ::=
    { sequential_statement }

sequential_statement ::=
    wait_statement
    | assertion_statement
    | report_statement
    | signal_assignment_statement
    | variable_assignment_statement
    | procedure_call_statement
    | if_statement
    | case_statement
    | loop_statement
    | next_statement
    | exit_statement
    | return_statement
    | null_statement

-- VHDL'93
shift_expression ::=
    simple_expression
    [ shift_operator simple_expression ]

-- VHDL'93
shift_operator ::= sll | srl | sla | sra | rol | ror

sign ::= + | -

-- VHDL'87
signal_assignment_statement ::=
    target <= [ transport ] waveform ;

-- VHDL'93
signal_assignment_statement ::=
    [ label : ] target <= [ delay_mechanism ]
        waveform ;

signal_declaration ::=
    signal identifier_list : subtype_indication
        [ signal_kind ] [ := expression ] ;

signal_kind ::= register | bus

signal_list ::=
    signal_name { , signal_name }
    | others
    | all

-- VHDL'93
signature ::= [ [ type_mark { , type_mark } ]
    [ return type_mark ] ]

simple_expression ::=
    [ sign ] term { adding_operator term }

simple_name ::= identifier

slice_name ::= prefix ( discrete_range )

string_literal ::= " { graphic_character } "

subprogram_body ::=
    subprogram_specification is
        subprogram_declarative_part
    begin
        subprogram_statement_part
    end [ subprogram_kind ] [ designator ] ;

subprogram_declaration ::=
    subprogram_specification ;

```

```

subprogram_declarative_item ::=
    subprogram_declaration
    | subprogram_body
    | type_declaration
    | subtype_declaration
    | constant_declaration
    | variable_declaration
    | file_declaration
    | alias_declaration
    | attribute_declaration
    | attribute_specification
    | use_clause
    | group_template_declaration
    | group_declaration

```

```

subprogram_declarative_part ::=
    { subprogram_declarative_item }

```

```

subprogram_kind ::= procedure | function

```

```

subprogram_specification ::=
    procedure designator [ ( { formal_parameter_list } ) ]
    | [ pure | impure ] function designator
      [ ( { formal_parameter_list } ) ]
      return type_mark

```

```

subprogram_statement_part ::=
    { sequential_statement }

```

```

subtype_declaration ::=
    subtype identifier is subtype_indication ;

```

```

subtype_indication ::=
    [ resolution_function_name ] type_mark
    [ constraint ]

```

```

suffix ::=
    simple_name
    | character_literal
    | operator_symbol
    | all

```

```

target ::=
    name
    | aggregate

```

```

term ::=
    factor { multiplying_operator factor }

```

```

timeout_clause ::= for time_expression

```

```

type_conversion ::= type_mark ( expression )

```

```

type_declaration ::=
    full_type_declaration
    | incomplete_type_declaration

```

```

type_definition ::=
    scalar_type_definition
    | composite_type_definition
    | access_type_definition
    | file_type_definition

```

```

type_mark ::=
    type_name
    | subtype_name

```

```

unconstrained_array_definition ::=
    array ( index_subtype_definition
           { , index_subtype_definition } )
           of element_subtype_indication

```

```

use_clause ::=
    use selected_name { , selected_name } ;

```

```

variable_assignment_statement ::=
    [ label : ] target := expression ;

```

```

variable_declaration ::=
    [ shared ] variable identifier_list :
    subtype_indication [ := expression ] ;

```

```

wait_statement ::=
    [ label : ] wait [ sensitivity_clause ]
    [ condition_clause ] [ timeout_clause ] ;

```

```

waveform ::=
    waveform_element { , waveform_element }
    | unaffected

```

```

waveform_element ::=
    value_expression [ after time_expression ]
    | null [ after time_expression ]

```

## Appendix B : PACKAGE STANDARD

```
-- This is Package STANDARD as defined in the VHDL 1992 Language Reference Manual.
-- Reprinted by permission from Model Technology Inc.
-- NOTE: VCOM and VSIM will not work properly if these declarations
-- are modified.

-- Version information: @(#)standard.vhd

package standard is
  type boolean is (false,true);
  type bit is ('0','1');
  type character is (
    nul, soh, stx, etx, eot, enq, ack, bel,
    bs, ht, lf, vt, ff, cr, so, si,
    dle, dc1, dc2, dc3, dc4, nak, syn, etb,
    can, em, sub, esc, fsp, gsp, rsp, usp,

    ' ', '!', '"', '#', '$', '%', '&', '\'',
    '(', ')', '*', '+', ',', '-', '.', '/',
    '0', '1', '2', '3', '4', '5', '6', '7',
    '8', '9', ':', ';', '<', '=', '>', '?',

    '@', 'A', 'B', 'C', 'D', 'E', 'F', 'G',
    'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O',
    'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W',
    'X', 'Y', 'Z', '[', '\', ']', '^', '_',

    '`', 'a', 'b', 'c', 'd', 'e', 'f', 'g',
    'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
    'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
    'x', 'y', 'z', '{', '|', '}', '~', del,

    c128, c129, c130, c131, c132, c133, c134, c135,
    c136, c137, c138, c139, c140, c141, c142, c143,
    c144, c145, c146, c147, c148, c149, c150, c151,
    c152, c153, c154, c155, c156, c157, c158, c159,

    -- the character code for 160 is there (NBSP),
    -- but prints as no char

    ' ', '!', '¢', '£', '¤', '¥', '¦', '§',
    '¨', '©', 'ª', «, ¬, ®, ¯,
    '°', '±', '²', '³', ´, µ, ¶, ¸,
    '¹', 'º', »', ¼, ½, ¾, ¿
```

```

        'À', 'Á', 'Â', 'Ã', 'Ä', 'Å', 'Æ', 'Ç',
        'È', 'É', 'Ê', 'Ë', 'Ì', 'Í', 'Î', 'Ï',
        'Ð', 'Ñ', 'Ò', 'Ó', 'Ô', 'Õ', 'Ö', '×',
        'Ø', 'Ù', 'Ú', 'Û', 'Ü', 'Ý', 'Þ', 'ß',

        'à', 'á', 'â', 'ã', 'ä', 'å', 'æ', 'ç',
        'è', 'é', 'ê', 'ë', 'ì', 'í', 'î', 'ï',
        'ð', 'ñ', 'ò', 'ó', 'ô', 'õ', 'ö', '÷',
        'ø', 'ù', 'ú', 'û', 'ü', 'ý', 'þ', 'ÿ' );

type severity_level is (note, warning, error, failure);
type integer is range -2147483647 to 2147483647; -- per LRM minimum range
type real is range -1.0E308 to 1.0E308;
type time is range -2147483647 to 2147483647 -- per LRM minimum range
    units
        fs;
        ps = 1000 fs;
        ns = 1000 ps;
        us = 1000 ns;
        ms = 1000 us;
        sec = 1000 ms;
        min = 60 sec;
        hr = 60 min;
    end units;
subtype delay_length is time range 0 fs to time'high;
impure function now return delay_length;
subtype natural is integer range 0 to integer'high;
subtype positive is integer range 1 to integer'high;
type string is array (positive range <>) of character;
type bit_vector is array (natural range <>) of bit;
type file_open_kind is (
    read_mode,
    write_mode,
    append_mode);
type file_open_status is (
    open_ok,
    status_error,
    name_error,
    mode_error);
attribute foreign : string;
end standard;

```

## Appendix C : PACKAGE TEXTIO

---

```
-----
-- Package TEXTIO as defined in Chapter 14 of the IEEE Standard VHDL
--   Language Reference Manual (IEEE Std. 1076-1987), as modified
--   by the Issues Screening and Analysis Committee (ISAC), a subcommittee
--   of the VHDL Analysis and Standardization Group (VASG) on
--   10 November, 1988.  See "The Sense of the VASG", October, 1989.
-- Reprinted by permission from Model Technology Inc.
-----
-- Version information: %W% %G%
-----

package TEXTIO is
  type LINE is access string;
  type TEXT is file of string;
  type SIDE is (right, left);
  subtype WIDTH is natural;

  -- changed for vhd192 syntax:
  file input : TEXT open read_mode is "STD_INPUT";
  file output : TEXT open write_mode is "STD_OUTPUT";

  -- changed for vhd192 syntax (and now a built-in):
  procedure READLINE(file f: TEXT; L: out LINE);

  procedure READ(L: inout LINE; VALUE: out bit; GOOD : out BOOLEAN);
  procedure READ(L: inout LINE; VALUE: out bit);

  procedure READ(L: inout LINE; VALUE: out bit_vector; GOOD : out BOOLEAN);
  procedure READ(L: inout LINE; VALUE: out bit_vector);

  procedure READ(L: inout LINE; VALUE: out BOOLEAN; GOOD : out BOOLEAN);
  procedure READ(L: inout LINE; VALUE: out BOOLEAN);

  procedure READ(L: inout LINE; VALUE: out character; GOOD : out BOOLEAN);
  procedure READ(L: inout LINE; VALUE: out character);

  procedure READ(L: inout LINE; VALUE: out integer; GOOD : out BOOLEAN);
  procedure READ(L: inout LINE; VALUE: out integer);

  procedure READ(L: inout LINE; VALUE: out real; GOOD : out BOOLEAN);
  procedure READ(L: inout LINE; VALUE: out real);

  procedure READ(L: inout LINE; VALUE: out string; GOOD : out BOOLEAN);
```

```

procedure READ(L:inout LINE; VALUE: out string);

procedure READ(L:inout LINE; VALUE: out time; GOOD : out BOOLEAN);
procedure READ(L:inout LINE; VALUE: out time);

-- changed for vhd192 syntax (and now a built-in):
procedure WRITELINE(file f : TEXT; L : inout LINE);

procedure WRITE(L : inout LINE; VALUE : in bit;
    JUSTIFIED: in SIDE := right;
    FIELD: in WIDTH := 0);

procedure WRITE(L : inout LINE; VALUE : in bit_vector;
    JUSTIFIED: in SIDE := right;
    FIELD: in WIDTH := 0);

procedure WRITE(L : inout LINE; VALUE : in BOOLEAN;
    JUSTIFIED: in SIDE := right;
    FIELD: in WIDTH := 0);

procedure WRITE(L : inout LINE; VALUE : in character;
    JUSTIFIED: in SIDE := right;
    FIELD: in WIDTH := 0);

procedure WRITE(L : inout LINE; VALUE : in integer;
    JUSTIFIED: in SIDE := right;
    FIELD: in WIDTH := 0);

procedure WRITE(L : inout LINE; VALUE : in real;
    JUSTIFIED: in SIDE := right;
    FIELD: in WIDTH := 0;
    DIGITS: in NATURAL := 0);

procedure WRITE(L : inout LINE; VALUE : in string;
    JUSTIFIED: in SIDE := right;
    FIELD: in WIDTH := 0);

procedure WRITE(L : inout LINE; VALUE : in time;
    JUSTIFIED: in SIDE := right;
    FIELD: in WIDTH := 0;
    UNIT: in TIME := ns);

-- is implicit built-in:
-- function ENDFILE(file F : TEXT) return boolean;

-- function ENDLINEL(variable L : in LINE) return BOOLEAN;
--
-- Function ENDLINEL as declared cannot be legal VHDL, and
-- the entire function was deleted from the definition
-- by the Issues Screening and Analysis Committee (ISAC),
-- a subcommittee of the VHDL Analysis and Standardization
-- Group (VASG) on 10 November, 1988. See "The Sense of
-- the VASG", October, 1989, VHDL Issue Number 0032.
end;

--*****
--**
--** Copyright (c) Model Technology Incorporated 1991 **
--** All Rights Reserved **
--**
--*****

```

## Appendix D : PACKAGE STD\_LOGIC\_1164

---

```
-- -----
--
-- Title      : std_logic_1164 multi-value logic system
-- Library    : This package shall be compiled into a library
--              : symbolically named IEEE.
--              :
-- Developers: IEEE model standards group (par 1164)
-- Purpose    : This packages defines a standard for designers
--              : to use in describing the interconnection data types
--              : used in vhdl modeling.
--              :
-- Limitation: The logic system defined in this package may
--              : be insufficient for modeling switched transistors,
--              : since such a requirement is out of the scope of this
--              : effort. Furthermore, mathematics, primitives,
--              : timing standards, etc. are considered orthogonal
--              : issues as it relates to this package and are therefore
--              : beyond the scope of this effort.
--              :
-- Note       : No declarations or definitions shall be included in,
--              : or excluded from this package. The "package declaration"
--              : defines the types, subtypes and declarations of
--              : std_logic_1164. The std_logic_1164 package body shall be
--              : considered the formal definition of the semantics of
--              : this package. Tool developers may choose to implement
--              : the package body in the most efficient manner available
--              : to them.
--              :
-- -----
-- modification history :
-- -----
-- version | mod. date:|
-- v4.200  | 01/02/92  |
-- -----

PACKAGE std_logic_1164 IS
-- logic state system (unresolved)
-- -----
TYPE std_ulogic IS ( 'U', -- Uninitialized
                      'X', -- Forcing Unknown
                      '0', -- Forcing 0
                      '1', -- Forcing 1
                      'Z', -- High Impedance
                      'W', -- Weak Unknown
```

```

        'L', -- Weak      0
        'H', -- Weak      1
        '-'  -- Don't care
    );

-----
-- unconstrained array of std_ulogic for use with the resolution function
-----
TYPE std_ulogic_vector IS ARRAY ( NATURAL RANGE <> ) OF std_ulogic;

-----
-- resolution function
-----
FUNCTION resolved ( s : std_ulogic_vector ) RETURN std_ulogic;

-----
-- *** industry standard logic type ***
-----
SUBTYPE std_logic IS resolved std_ulogic;

-----
-- unconstrained array of std_logic for use in declaring signal arrays
-----
TYPE std_logic_vector IS ARRAY ( NATURAL RANGE <> ) OF std_logic;

-----
-- common subtypes
-----
SUBTYPE X01      IS resolved std_ulogic RANGE 'X' TO '1'; -- ('X','0','1')
SUBTYPE X01Z    IS resolved std_ulogic RANGE 'X' TO 'Z'; -- ('X','0','1','Z')
SUBTYPE UX01    IS resolved std_ulogic RANGE 'U' TO '1'; -- ('U','X','0','1')
SUBTYPE UX01Z   IS resolved std_ulogic RANGE 'U' TO 'Z';
    -- ('U','X','0','1','Z')

-----
-- overloaded logical operators
-----

FUNCTION "and" ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
FUNCTION "nand" ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
FUNCTION "or"  ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
FUNCTION "nor" ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
FUNCTION "xor" ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
FUNCTION "xnor" ( l : std_ulogic; r : std_ulogic ) RETURN ux01;
FUNCTION "not" ( l : std_ulogic ) RETURN UX01;

-----
-- vectorized overloaded logical operators
-----
FUNCTION "and" ( l, r : std_logic_vector ) RETURN std_logic_vector;
FUNCTION "and" ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;

FUNCTION "nand" ( l, r : std_logic_vector ) RETURN std_logic_vector;
FUNCTION "nand" ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;

FUNCTION "or"  ( l, r : std_logic_vector ) RETURN std_logic_vector;
FUNCTION "or"  ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;

FUNCTION "nor" ( l, r : std_logic_vector ) RETURN std_logic_vector;
FUNCTION "nor" ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;

FUNCTION "xor" ( l, r : std_logic_vector ) RETURN std_logic_vector;

```



```

FUNCTION "xor" ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;
-----
-- Note : The declaration and implementation of the "xnor" function is
-- specifically commented until at which time the VHDL language has been
-- officially adopted as containing such a function. At such a point,
-- the following comments may be removed along with this notice without
-- further "official" balloting of this std_logic_1164 package. It is
-- the intent of this effort to provide such a function once it becomes
-- available in the VHDL standard.
-----
-- function "xnor" ( l, r : std_logic_vector ) return std_logic_vector;
-- function "xnor" ( l, r : std_ulogic_vector ) return std_ulogic_vector;

FUNCTION "not" ( l : std_logic_vector ) RETURN std_logic_vector;
FUNCTION "not" ( l : std_ulogic_vector ) RETURN std_ulogic_vector;

-----
-- conversion functions
-----
FUNCTION To_bit ( s : std_ulogic; xmap : BIT := '0') RETURN BIT;
FUNCTION To_bitvector ( s : std_logic_vector ; xmap : BIT := '0')
RETURN BIT_VECTOR;
FUNCTION To_bitvector ( s : std_ulogic_vector; xmap : BIT := '0')
RETURN BIT_VECTOR;

FUNCTION To_StdULogic ( b : BIT ) RETURN std_ulogic;
FUNCTION To_StdLogicVector ( b : BIT_VECTOR ) RETURN std_logic_vector;
FUNCTION To_StdLogicVector ( s : std_ulogic_vector ) RETURN std_logic_vector;
FUNCTION To_StdULogicVector ( b : BIT_VECTOR ) RETURN std_ulogic_vector;
FUNCTION To_StdULogicVector ( s : std_ulogic_vector) RETURN std_ulogic_vector;

-----
-- strength strippers and type convertors
-----
FUNCTION To_X01 ( s : std_logic_vector ) RETURN std_logic_vector;
FUNCTION To_X01 ( s : std_ulogic_vector ) RETURN std_ulogic_vector;
FUNCTION To_X01 ( s : std_ulogic ) RETURN X01;
FUNCTION To_X01 ( b : BIT_VECTOR ) RETURN std_logic_vector;
FUNCTION To_X01 ( b : BIT_VECTOR ) RETURN std_ulogic_vector;
FUNCTION To_X01 ( b : BIT ) RETURN X01;

FUNCTION To_X01Z ( s : std_logic_vector ) RETURN std_logic_vector;
FUNCTION To_X01Z ( s : std_ulogic_vector ) RETURN std_ulogic_vector;
FUNCTION To_X01Z ( s : std_ulogic ) RETURN X01Z;
FUNCTION To_X01Z ( b : BIT_VECTOR ) RETURN std_logic_vector;
FUNCTION To_X01Z ( b : BIT_VECTOR ) RETURN std_ulogic_vector;
FUNCTION To_X01Z ( b : BIT ) RETURN X01Z;

FUNCTION To_UX01 ( s : std_logic_vector ) RETURN std_logic_vector;
FUNCTION To_UX01 ( s : std_ulogic_vector ) RETURN std_ulogic_vector;
FUNCTION To_UX01 ( s : std_ulogic ) RETURN UX01;
FUNCTION To_UX01 ( b : BIT_VECTOR ) RETURN std_logic_vector;
FUNCTION To_UX01 ( b : BIT_VECTOR ) RETURN std_ulogic_vector;
FUNCTION To_UX01 ( b : BIT ) RETURN UX01;

-----
-- edge detection
-----
FUNCTION rising_edge (SIGNAL s : std_ulogic) RETURN BOOLEAN;
FUNCTION falling_edge (SIGNAL s : std_ulogic) RETURN BOOLEAN;

-----
-- object contains an unknown
-----
FUNCTION Is_X ( s : std_ulogic_vector ) RETURN BOOLEAN;
FUNCTION Is_X ( s : std_logic_vector ) RETURN BOOLEAN;
FUNCTION Is_X ( s : std_ulogic ) RETURN BOOLEAN;
END std_logic_1164;

```

## **Appendix E : PACKAGE STD\_LOGIC\_ARITH**

---

```
-- Copyright (c) 1990, 1991, 1992 by Synopsys, Inc. All rights reserved. --
--
-- This source file may be used and distributed without restriction --
-- provided that this copyright statement is not removed from the file --
-- and that any derivative work contains this copyright notice. --
--
-- Package name: STD_LOGIC_ARITH --
--
-- Purpose: --
-- A set of arithmetic, conversion, and comparison functions --
-- for SIGNED, UNSIGNED, SMALL_INT, INTEGER, --
-- STD_ULONGIC, STD_LOGIC, and STD_LOGIC_VECTOR. --
--
-----
```

```
library IEEE;
use IEEE.std_logic_1164.all;
```

```
package std_logic_arith is
```

```
  type UNSIGNED is array (NATURAL range <>) of STD_LOGIC;
  type SIGNED is array (NATURAL range <>) of STD_LOGIC;
  subtype SMALL_INT is INTEGER range 0 to 1;
```

<b>function</b> "+"(L: UNSIGNED; R: UNSIGNED)	<b>return</b> UNSIGNED;
<b>function</b> "+"(L: SIGNED; R: SIGNED)	<b>return</b> SIGNED;
<b>function</b> "+"(L: UNSIGNED; R: SIGNED)	<b>return</b> SIGNED;
<b>function</b> "+"(L: SIGNED; R: UNSIGNED)	<b>return</b> SIGNED;
<b>function</b> "+"(L: UNSIGNED; R: INTEGER)	<b>return</b> UNSIGNED;
<b>function</b> "+"(L: INTEGER; R: UNSIGNED)	<b>return</b> UNSIGNED;
<b>function</b> "+"(L: SIGNED; R: INTEGER)	<b>return</b> SIGNED;
<b>function</b> "+"(L: INTEGER; R: SIGNED)	<b>return</b> SIGNED;

<b>function</b> "+"(L: UNSIGNED; R: STD_ULOGIC)	<b>return</b> UNSIGNED;
<b>function</b> "+"(L: STD_ULOGIC; R: UNSIGNED)	<b>return</b> UNSIGNED;
<b>function</b> "+"(L: SIGNED; R: STD_ULOGIC)	<b>return</b> SIGNED;
<b>function</b> "+"(L: STD_ULOGIC; R: SIGNED)	<b>return</b> SIGNED;
<b>function</b> "+"(L: UNSIGNED; R: UNSIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: SIGNED; R: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: UNSIGNED; R: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: SIGNED; R: UNSIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: UNSIGNED; R: INTEGER)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: INTEGER; R: UNSIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: SIGNED; R: INTEGER)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: INTEGER; R: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: UNSIGNED; R: STD_ULOGIC)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: STD_ULOGIC; R: UNSIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: SIGNED; R: STD_ULOGIC)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: STD_ULOGIC; R: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "-"(L: UNSIGNED; R: UNSIGNED)	<b>return</b> UNSIGNED;
<b>function</b> "-"(L: SIGNED; R: SIGNED)	<b>return</b> SIGNED;
<b>function</b> "-"(L: UNSIGNED; R: SIGNED)	<b>return</b> SIGNED;
<b>function</b> "-"(L: SIGNED; R: UNSIGNED)	<b>return</b> SIGNED;
<b>function</b> "-"(L: UNSIGNED; R: INTEGER)	<b>return</b> UNSIGNED;
<b>function</b> "-"(L: INTEGER; R: UNSIGNED)	<b>return</b> UNSIGNED;
<b>function</b> "-"(L: SIGNED; R: INTEGER)	<b>return</b> SIGNED;
<b>function</b> "-"(L: INTEGER; R: SIGNED)	<b>return</b> SIGNED;
<b>function</b> "-"(L: UNSIGNED; R: STD_ULOGIC)	<b>return</b> UNSIGNED;
<b>function</b> "-"(L: STD_ULOGIC; R: UNSIGNED)	<b>return</b> UNSIGNED;
<b>function</b> "-"(L: SIGNED; R: STD_ULOGIC)	<b>return</b> SIGNED;
<b>function</b> "-"(L: STD_ULOGIC; R: SIGNED)	<b>return</b> SIGNED;
<b>function</b> "-"(L: UNSIGNED; R: UNSIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "-"(L: SIGNED; R: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "-"(L: UNSIGNED; R: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "-"(L: SIGNED; R: UNSIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "-"(L: UNSIGNED; R: INTEGER)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "-"(L: INTEGER; R: UNSIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "-"(L: SIGNED; R: INTEGER)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "-"(L: INTEGER; R: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "-"(L: UNSIGNED; R: STD_ULOGIC)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "-"(L: STD_ULOGIC; R: UNSIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "-"(L: SIGNED; R: STD_ULOGIC)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "-"(L: STD_ULOGIC; R: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: UNSIGNED)	<b>return</b> UNSIGNED;
<b>function</b> "+"(L: SIGNED)	<b>return</b> SIGNED;
<b>function</b> "-"(L: SIGNED)	<b>return</b> SIGNED;
<b>function</b> "ABS"(L: SIGNED)	<b>return</b> SIGNED;
<b>function</b> "+"(L: UNSIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "+"(L: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;

<b>function</b> "-"(L: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "ABS"(L: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "**"(L: UNSIGNED; R: UNSIGNED)	<b>return</b> UNSIGNED;
<b>function</b> "**"(L: SIGNED; R: SIGNED)	<b>return</b> SIGNED;
<b>function</b> "**"(L: SIGNED; R: UNSIGNED)	<b>return</b> SIGNED;
<b>function</b> "**"(L: UNSIGNED; R: SIGNED)	<b>return</b> SIGNED;
<b>function</b> "**"(L: UNSIGNED; R: UNSIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "**"(L: SIGNED; R: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "**"(L: SIGNED; R: UNSIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "**"(L: UNSIGNED; R: SIGNED)	<b>return</b> STD_LOGIC_VECTOR;
<b>function</b> "<"(L: UNSIGNED; R: UNSIGNED)	<b>return</b> BOOLEAN;
<b>function</b> "<"(L: SIGNED; R: SIGNED)	<b>return</b> BOOLEAN;
<b>function</b> "<"(L: UNSIGNED; R: SIGNED)	<b>return</b> BOOLEAN;
<b>function</b> "<"(L: SIGNED; R: UNSIGNED)	<b>return</b> BOOLEAN;
<b>function</b> "<"(L: UNSIGNED; R: INTEGER)	<b>return</b> BOOLEAN;
<b>function</b> "<"(L: INTEGER; R: UNSIGNED)	<b>return</b> BOOLEAN;
<b>function</b> "<"(L: SIGNED; R: INTEGER)	<b>return</b> BOOLEAN;
<b>function</b> "<"(L: INTEGER; R: SIGNED)	<b>return</b> BOOLEAN;
<b>function</b> "<="(L: UNSIGNED; R: UNSIGNED)	<b>return</b> BOOLEAN;
<b>function</b> "<="(L: SIGNED; R: SIGNED)	<b>return</b> BOOLEAN;
<b>function</b> "<="(L: UNSIGNED; R: SIGNED)	<b>return</b> BOOLEAN;
<b>function</b> "<="(L: SIGNED; R: UNSIGNED)	<b>return</b> BOOLEAN;
<b>function</b> "<="(L: UNSIGNED; R: INTEGER)	<b>return</b> BOOLEAN;
<b>function</b> "<="(L: INTEGER; R: UNSIGNED)	<b>return</b> BOOLEAN;
<b>function</b> "<="(L: SIGNED; R: INTEGER)	<b>return</b> BOOLEAN;
<b>function</b> "<="(L: INTEGER; R: SIGNED)	<b>return</b> BOOLEAN;
<b>function</b> ">"(L: UNSIGNED; R: UNSIGNED)	<b>return</b> BOOLEAN;
<b>function</b> ">"(L: SIGNED; R: SIGNED)	<b>return</b> BOOLEAN;
<b>function</b> ">"(L: UNSIGNED; R: SIGNED)	<b>return</b> BOOLEAN;
<b>function</b> ">"(L: SIGNED; R: UNSIGNED)	<b>return</b> BOOLEAN;
<b>function</b> ">"(L: UNSIGNED; R: INTEGER)	<b>return</b> BOOLEAN;
<b>function</b> ">"(L: INTEGER; R: UNSIGNED)	<b>return</b> BOOLEAN;
<b>function</b> ">"(L: SIGNED; R: INTEGER)	<b>return</b> BOOLEAN;
<b>function</b> ">"(L: INTEGER; R: SIGNED)	<b>return</b> BOOLEAN;
<b>function</b> ">="(L: UNSIGNED; R: UNSIGNED)	<b>return</b> BOOLEAN;
<b>function</b> ">="(L: SIGNED; R: SIGNED)	<b>return</b> BOOLEAN;
<b>function</b> ">="(L: UNSIGNED; R: SIGNED)	<b>return</b> BOOLEAN;
<b>function</b> ">="(L: SIGNED; R: UNSIGNED)	<b>return</b> BOOLEAN;
<b>function</b> ">="(L: UNSIGNED; R: INTEGER)	<b>return</b> BOOLEAN;
<b>function</b> ">="(L: INTEGER; R: UNSIGNED)	<b>return</b> BOOLEAN;
<b>function</b> ">="(L: SIGNED; R: INTEGER)	<b>return</b> BOOLEAN;
<b>function</b> ">="(L: INTEGER; R: SIGNED)	<b>return</b> BOOLEAN;
<b>function</b> "="(L: UNSIGNED; R: UNSIGNED)	<b>return</b> BOOLEAN;

```

function "="(L: SIGNED; R: SIGNED)           return BOOLEAN;
function "="(L: UNSIGNED; R: SIGNED)         return BOOLEAN;
function "="(L: SIGNED; R: UNSIGNED)         return BOOLEAN;
function "="(L: UNSIGNED; R: INTEGER)        return BOOLEAN;
function "="(L: INTEGER; R: UNSIGNED)        return BOOLEAN;
function "="(L: SIGNED; R: INTEGER)          return BOOLEAN;
function "="(L: INTEGER; R: SIGNED)          return BOOLEAN;

function "/="(L: UNSIGNED; R: UNSIGNED)       return BOOLEAN;
function "/="(L: SIGNED; R: SIGNED)           return BOOLEAN;
function "/="(L: UNSIGNED; R: SIGNED)         return BOOLEAN;
function "/="(L: SIGNED; R: UNSIGNED)         return BOOLEAN;
function "/="(L: UNSIGNED; R: INTEGER)        return BOOLEAN;
function "/="(L: INTEGER; R: UNSIGNED)        return BOOLEAN;
function "/="(L: SIGNED; R: INTEGER)          return BOOLEAN;
function "/="(L: INTEGER; R: SIGNED)          return BOOLEAN;

function SHL(ARG: UNSIGNED; COUNT: UNSIGNED) return UNSIGNED;
function SHL(ARG: SIGNED; COUNT: UNSIGNED)   return SIGNED;
function SHR(ARG: UNSIGNED; COUNT: UNSIGNED) return UNSIGNED;
function SHR(ARG: SIGNED; COUNT: UNSIGNED)   return SIGNED;

function CONV_INTEGER(ARG: INTEGER)          return INTEGER;
function CONV_INTEGER(ARG: UNSIGNED)         return INTEGER;
function CONV_INTEGER(ARG: SIGNED)           return INTEGER;
function CONV_INTEGER(ARG: STD_ULOGIC)       return SMALL_INT;

function CONV_UNSIGNED(ARG: INTEGER; SIZE: INTEGER) return UNSIGNED;
function CONV_UNSIGNED(ARG: UNSIGNED; SIZE: INTEGER) return UNSIGNED;
function CONV_UNSIGNED(ARG: SIGNED; SIZE: INTEGER) return UNSIGNED;
function CONV_UNSIGNED(ARG: STD_ULOGIC; SIZE: INTEGER) return UNSIGNED;

function CONV_SIGNED(ARG: INTEGER; SIZE: INTEGER) return SIGNED;
function CONV_SIGNED(ARG: UNSIGNED; SIZE: INTEGER) return SIGNED;
function CONV_SIGNED(ARG: SIGNED; SIZE: INTEGER) return SIGNED;
function CONV_SIGNED(ARG: STD_ULOGIC; SIZE: INTEGER) return SIGNED;

function CONV_STD_LOGIC_VECTOR(ARG: INTEGER; SIZE: INTEGER)
                                     return STD_LOGIC_VECTOR;
function CONV_STD_LOGIC_VECTOR(ARG: UNSIGNED; SIZE: INTEGER)
                                     return STD_LOGIC_VECTOR;
function CONV_STD_LOGIC_VECTOR(ARG: SIGNED; SIZE: INTEGER)
                                     return STD_LOGIC_VECTOR;
function CONV_STD_LOGIC_VECTOR(ARG: STD_ULOGIC; SIZE: INTEGER)
                                     return STD_LOGIC_VECTOR;

```

-- zero extend STD\_LOGIC\_VECTOR (ARG) to SIZE,

```
-- SIZE < 0 is same as SIZE = 0
-- returns STD_LOGIC_VECTOR(SIZE-1 downto 0)
function EXT(ARG: STD_LOGIC_VECTOR; SIZE: INTEGER) return STD_LOGIC_VECTOR;

-- sign extend STD_LOGIC_VECTOR (ARG) to SIZE,
-- SIZE < 0 is same as SIZE = 0
-- return STD_LOGIC_VECTOR(SIZE-1 downto 0)
function SXT(ARG: STD_LOGIC_VECTOR; SIZE: INTEGER) return STD_LOGIC_VECTOR;

end Std_logic_arith;
```

## Appendix F :VHDL PREDEFINED ATTRIBUTES

---



---

### VHDL Attributes

Attribute	Prefix	Comments
T'base	Type	Base type of T. Must be prefix to another attribute
T'left	scalar	The left bound of T, result of type T
T'right	type/ST	The right bound of T, result of type T
T'high	scalar	The upper bound of T, result of type T
T'low	type/ST	The lower bound of T, result of type T
T'Ascending VHDL'93	scalar type/ST	TRUE if type T is ascending
T'image(X) VHDL'93	scalar type/ST	Function which converts scalar object X of type T into string
T'value(X) VHDL'93	scalar type/ST	Function which converts object X of type string into scalar of type T
T'pos(X)	discrete /PT/ST	Function which returns a universal integer representing the position number of parameter X of type T. First position = 0.
T'val(X)	discrete /PT/ST	Function which returns of base type T the value whose position is the universal integer value corresponding to X.
T'succ(X)	discrete /PT/ST	Function returning a value of type T whose value is the position number one greater than the one of the parameter. It is an error if X = T'high or if does not belong to the range T'low to T'high
T'pred(X)	discrete /PT/ST	Function returning a value of type T whose value is the position number one less than the one of the parameter. It is an error if X = T'low or if does not belong to the range T'low to T'high
T'leftof(X)	discrete /PT/ST	Function which returns the value that is to the left of parameter X of type T. Result type is of type T. Error if X = T'left

T'rightof(X)	discrete /PT/ST	Function which returns the value that is to the right of parameter X of type T. Result type is of type T. Error is X = T'right
A'left(N)	Array*	Function which returns the left bound of of the Nth index range of A. X is of type universal integer. Result type is of type of the left bound of the left index range of A. N = 1 if omitted.
A'right(A)	Array*	Same as A'left(N), except right bound is returned
A'high(N)	Array*	Function which returns the upper bound of the range of A. Result type is the type of the Nth index range of A. N = 1 if omitted.
A'low(N)	Array*	Same as A'high(N), e lower bound is returned.
A'range(N)	Array*	The range of A'left(N) to A'right(N)
A'reverse range(N)		The range of A'right(N) to A'left(N)
A'length	Array*	returns 0 if array is null. Else, returns T'pos(A'high(N)) - T'pos(A'low(N)) where T is the subtype of the Nth index of A.
A'Ascending	Array*	True if Nth index range of A is defined in an ascending range, else returns false.

PT = physical type

ST = Subtype

Array\* = Any prefix that is appropriate for an array object, (e.g. type, variable, signal) or alias thereof, or that denotes a constrained array subtype

### Summary of the VHDL Signal Attributes

S'event	Function returning a <b>Boolean</b> which identifies if signal S has a new value assigned onto this signal (i.e. value is different that last value). if Clk'event then -- if Clk just changed in value then ... .... <b>wait until Clk'event and Clk = '1';</b> -- rising edge of clock
S'active	Function returning a <b>Boolean</b> which identifies if signal S had a new assignment made onto it (whether the value of the assignment is the SAME or DIFFERENT). if Data'active then -- New assignment of Data
S'transaction	<b>Implicit signal</b> of type bit which is created for signal S when it S'transaction is used in the code. This implicit signal is NOT declared since it is implicitly defined. This signal toggles in value (between '0' and '1') when signal S had a new assignment made onto it (whether the value of the assignment is the SAME or DIFFERENT. The user should NOT rely on its VALUE. <b>wait on ReceivedData'transaction;</b> -- process is -- sensitive to ReceiveData rewriting any value



S'delayed(T)	<p><b>Implicit signal</b> of the same base type as S. It represents the value of signal S delayed by a time Tn. Thus, the value of S'delayed(T) at time Tn is always equal to the value of S at time Tn -t. For example, the value of S'delayed(5 ns) at time 1000 ns is the value of S at time 995 ns. Note if time is omitted, it defaults to 0 ns.</p> <pre> wait on Data'transaction; case BV2'(Data'Delayed &amp; Data) is -- Data @ last delta time   when "X0" =&gt; ... -- from X to 0 transition   when "10" =&gt; ... -- from 1 to 0 transition   when others =&gt; ... -- end case; </pre>
S'stable(T)	<p><b>Implicit signal</b> of Boolean type. This implicit signal is true when an event (change in value) has NOT occurred on signal S for T time units, and the value FALSE otherwise. If time is omitted, it defaults to 0 ns.</p> <pre> if Data'stable(40 ns) then -- met set up time </pre>
S'quiet(T)	<p><b>Implicit signal</b> of Boolean type. This implicit signal is true when the signal has been quiet (i.e. no activity or signal assignment) for T time units, and the value FALSE otherwise. If time is omitted, it defaults to 0 ns.</p> <pre> if Data'quiet(40 ns) then -- Really quiet, not even an assignment of -- the same value during the last T time units (40 ns in this example) </pre>
S'last_event	<p>The amount of time that has elapsed since the last event (change in value) occurred on signal S. If there was no previous event, it returns Time'high.</p> <pre> variable : TsinceLastEvent : time; -- .. TsinceLastEvent := Data'last_event; </pre>
S'last_active	<p>The amount of time that has elapsed since the last activity (assignment) occurred on signal S. If there was no previous activity, it returns Time'high.</p> <pre> variable : TsinceLastActivity : time; -- .. TsinceLastActivity := Data'last_active; </pre>
S'last_value	<p>Function of the base type of S returning the previous value of S, immediately before the last change of S.</p> <pre> wait on Data'transaction; case BV2'(Data'last_value &amp; Data) is   when "X0" =&gt; ... -- from X to 0 transition   when "10" =&gt; ... -- from 1 to 0 transition   when "00" =&gt; .. -- Activity, but no Data change   when others =&gt; ... -- end case; </pre>

# **BIBLIOGRAPHY**

---

- [1] **IEEE**, *IEEE Std 1076-1993 IEEE Standard VHDL Language Reference Manual*, Copyright © 1994, Institute of Electrical and Electronics Engineers, Inc. Web site: <http://stdsbbs.ieee.org/>
- [2] **Sandi Habinc**, *VHDL Models for Board-Level Simulation*, European Space Agency, 1996, available via *ftp* from <ftp.estec.esa.nl/pub/vhdl/doc/BoardLevel.ps>
- [3] **Cadence Design Systems, Inc.**, HDL Design Group, 270 Billerica Road, Chelmsford MA 01824, Web site: <http://www.cadence.com/>
- [4] **Alternative System Concepts, Inc.**, PO Box 128 Windham NH 03087, email: [info@ascinc.com](mailto:info@ascinc.com)
- [5] **Scientific and Engineering Software, Inc.**, SES Workbench system simulator, 4301 Westbank Drive, Blg. A, Austin, TX 78746, Web site: <http://www.ses.com>
- [6] **Model Technology Incorporated**, Vendor of *Vsystem* VHDL simulator for PC and workstations, 8905 SW Nimbus Av., Suite 150. Beaverton OR 97008-7100, Web site: <http://www.model.com>
- [7] **David Naiditch**, *RendezVous with Ada 95*, John Wiley and Sons Inc. Copyright © 1995 by David J. Naiditch, Reprinted by permission of John Wiley & Sons, Inc.
- [8] **Daniel S. Barclay**, Compass Design Automation, Inc.
- [9] **Ben Cohen**, *VHDL Coding Styles and Methodologies*, ISBN 0-7923-9598-0 Kluwer Academic Publishers, 1995, Web site: [members.aol.com/vhdlcohen](http://members.aol.com/vhdlcohen)
- [10] **Paul H. Bardell, William H. McAnney, Jacob Savir**, *Built-in Test for VLSI: Pseudorandom Techniques*, John Wiley and Sons, 1987.
- [11] **Joseph Pick** of Synopsys, Inc., *VHDL Synthesis Techniques and Recommendations*, presented at the 1996 Synopsys Users Group Conference
- [12] **QuickLogic Corporation**, 2933 Bunker Hill Lane, Santa Clara, CA 95054, email: [info@qlogic.com](mailto:info@qlogic.com)
- [13] **Synplicity, Inc.**, 465 Fairchild Drive, Suite 115, Mountain View, CA 94043
- [14] **Synopsys**, *VHDL Compiler Reference Manual*, Document Order Number: IUS01-10430, Web site: <http://www.synopsys.com>
- [15] **COMPASS Design Automation**, 5457 Twin Knolls Road, Suite 100, Columbia, MD 21045, Web site: <http://www.compass-da.com>
- [16] **S. Habinc and P. Sinander**, *Using VHDL for Board Level Simulation*, IEEE Design & Test of Computers, Vol. 13, No. 3, Fall 1996, pp. 66-78; Web site: <http://www.computer.org/pubs/d&t/d&t.htm>
- [17] **Chronology**, *Timing Designer* and *QuickBench*, [sales@chronology.com](mailto:sales@chronology.com), URL: <http://www.chronology.com>, (206)869-4227.

# INDEX

---

## A

### Access Type

- Application example, 154
- Example, 107
- Garbage collection, 106
- Linked list
  - Example, 154

### Activity

- Resolved signal, 85

### Aggregate

- Synthesis, 185
- Choices, 303

### Alias

- disk\element\cntl\_r\_e.vhd, 45
- Fields
  - Std\_Logic\_Vector, 44
- Synthesis, 184

### Array

- Aggregate, 57, 75
- Aggregate, 74, 76
- Allowed objects, 73
- Anonymous, 78
- Assignment, 58
- Concatenation, 58
- Constrained, 71
- Constraining methods, 72
- Constraining methods, 79
- Definition, 71
- Equality, 58
- File
  - Emulation, 114
- Illegal declarations, 78
- Inequality, 58
- Initialization
  - Error, 67
- Logical operators, 59
- Mapping, 74
  - disk\element\mutidm.vhd, 75
  - disk\element\multidm.vhd, 74
- Multidimensional
  - Mapping, 74
- Normalization, 118
- Null, 65
  - Synthesis, 206
- One-dimensional, 56
- Operations
  - Overloaded, 60
- Operations
  - disk\element\array1.vhd, 59

### Operators

- Implicit, 58
  - disk\element\signed.vhd, 61
- Others, 76
- Range definition, 303
- Relational, 58
- Resolution function
  - disk\drivers\atomc\_pb.vhd, 86
- Shift operators, 59
- Slice, 65
- Slicing, 58
- Structure representations, 56, 186
- Synthesis, 73
- Table lookup, 56
- Two-dimensional, 56
  - disk\element\tmspec.vhd, 57
- type, 71
- Type conversion, 59
- Unconstrained, 71
  - Illegal, 78
  - Others
    - Disk\element\unconsok.vhd, 76

### Asynchronous reset

- Synthesis, 191

### Atomicity

- Drivers, 84

### Attribute

- 'Val, 47
- 'Image, 126
- 'Quiet
  - Application, 161
- 'transaction
  - Application, 159
- 'Val
  - disk\element\attribte.vhd, 48
- Enum\_Encoding, 47
- Exam, 302, 304, 328
- Image, 122, 126
- predefined (LRM 14.1), 302
- Reuse, 302, 328
- Synthesis
  - Supported/Unsupported, 185

### Auto-Regression

- See Regression/Verification, 246

## B

### Barrel shifter

- Synthesis/Behavioral, 205

### Behavioral Synthesis, 243

**Behavioral/RTL Modeling, 296****BFM (Bus Functional Model)**

- Architectural command, 255
- Definition, 254
- Instruction file command, 255
- Memory, 146
- Verification
  - Application, 254

**Binding, 24**

- Default, 25

**Bit string literal, 44****Block**

- Example, 11
- Reuse, 323
- Synthesis, 323
  - Advantages, 10
  - Restrictions, 185
- Statement
  - Guarded in Synthesis, 17
- Salient points, 10
- Synthesis, 10
- Reusability, 323

**Buffer Port**

- disk\element\buffer2.vhd, 18
- disk\element\bufftop2.vhd, 19
- disk\element\bufftop3.vhd, 19
- InOut Port, 17

**C****Case, 209**

- & operator., 42
- Choice, 70
- Globally static
  - restrictions, 319
- Priority, 209
- Don't care, 207

**Casting, 39, 44****Client/server**

- disk\subprgm\drvproc.vhd, 105
- Side Effect
  - Avoidance method, 103

**Code**

- Non-portable, 92

**Coding style**

- for design reuse, 316

**Compilation order, 298****Component**

- Removal, 31
- Salient points, 8

**Concatenation, 43**

- Procedure call, 230

**Concurrent Assertion**

- Salient Points, 9

**Concurrent procedure**

- Salient points, 8
- Side effect, 8

**Concurrent Signal assignment**

- for Waveforms, 8
- Salient points, 7

**Concurrent statement**

- Simulation speed, 273
- Synthesis., 185
- Definitions, 6
- Exam, 309

**Conditional Signal Assignment**

- Example, 7

**Configuration**

- Binding with configured components, 31
- Configuration Declaration, 28
  - declaration, 48
    - Configured Components
      - disk\element\struct3.vhd, 31
    - Deferred Binding
      - disk\element\deferd\_c.vhd, 32
    - disk\element\cfgdecl.vhd, 29
    - disk\element\cntl\_r\_c.vhd, 49
    - Generate statement
      - disk\element\generat\_c.vhd, 35
    - Generate statements, 33
    - Generics, 32
      - disk\element\filter2.vhd, 33
    - Hierarchical
      - disk\element\hierarch.vhd, 30
  - Declaration/Specification, 25
  - Specification
    - disk\element\cfgspec.vhd, 27
    - Requirements, 24
    - Synthesis, 185

**Constant**

- disk\element\constant.vhd, 45
- Fields
  - Std\_Logic\_Vector, 44
- Simulation speed, 270

**Constructs**

- Synthesis
  - Supported/Unsupported, 184

**Control**

- Parameterization, 319

**Conversion**

- Hexadecimal, 122
- Integer to time, 116
- Type, 39
- Typed objects to string, 122

**Counter**

- Design
  - Exam, 306
- disk
  - element\count\_ea.vhd, 41
- Model, 287

**D****Data reduction**

- MISR/Verifier, 130

**Deallocate, 106**

- disk\subprgm\accessok.vhd, 109
- disk\subprgm\image.vhd, 107

**Declaration**

- Illegal reference, 67

**Delta time**

- Restrictions on model, 162

**Demultiplexer**

- Synthesis, 203

**Design**

- Processes, 314
- Units
  - Exam, 297
- Verification, 246, 248, 251

**DesignWare**

- Reuse, 329
- Synthesis:, 234

**Disconnect, 17****Documentation, 337**

- Applicable documents, 331
- Application notes, 335
- Definitions, 331
- Design description, 332
- Design files, 335
- Design team, 331
- Designs, 331
- Diagram, 333
- Electrical specification, 335
- Initialization, 335
- Interface, 332
- ISA, 334
- Mapping and routing, 336
- Modes, 333
- Rationale, 333
- Requirement/design matrix, 332
- Resuability, 331
- Restrictions, 334
- Scaleability, 334
- Simulation, 338
- Software model, 334
- Summary, 331
- Synthesis, 336
- Testability, 334
- Testbench, 338
- Timing specification, 335
- Verification, 337

**Don't care**

- Numeric\_Std
  - Std\_Match, 211
- Synthesis
  - Guideline, 207, 212

**Drivers**

- Atomicity, 84
- Creation, 82
- Exam, 309
- Initial value, 83
- Initialization, 83
- LRM 12.6.1, 83
- LRM 4.3.1.2, 84
- Multiple, 82
  - Composite, 91
  - Source, 93
  - Static, 92
- Procedure, 100
- Resolved signal, 85
- Static, 82, 83
- Static expression, 92
- Subelement, 82

## E

**EDIF**

- Synthesis, 336

**Editor**

- VHDL, 294

**Elements**

- Inter-relationships, 2, 3

**Encoding**

- One-hot, 43

**Entity**

- Synthesis, 184

**Enumeration**

- Conversion
  - disk\element\convsynt.vhd, 47
- Encoding, 43
- Fields, 46
- Synthesis, 47

**Error**

- 'Image, 129
- Alias
  - Synthesis, 119
- Array
  - Anonymous, 78
- Array
  - Initialization, 67
  - Null slice, 66
  - Unconstrained, 78
- Association
  - Ground/VCC, 239
- Buffer port, 19
- Case
  - Choice, 70
  - Expression, 42
- Concatenation operator, 42
- Deallocation, 106, 107
- Declaration
  - Identifier, 67
- Drivers
  - Multiple, 82, 91, 92, 93
- Function parameters, 110
- Incompatible array types, 72
- Inertial signal assignment, 227
- Integer
  - Range, 38
- Others
  - Unconstrained array, 75

**Slice**

- Null, 66

- Std\_Logic\_1164 package '87/'93, 43

**Subprogram**

- Normalization, 118
- Signal attributes, 110

- Subprogram overloading, 64

**Synthesis**

- Dynamic value, 206
- Initialization, 194, 195
- Null slice, 205

**Type declaration**

- Multiple declarations, 293

- Types, 39

- Unconstrained Array, 78

**Error Injector, 163**

- Applications, 164
- Modes, 166

**Exam**

- Aggregate choices, 303
- Attributes, 302
- Concurrent Statements, 309
- Control structure, 303
- Counter design, 306
- Design units, 297
- Drivers, 309
- Final, 297
- Operator overloading, 306
- Resolution function
  - Boolean, 309
- Signal/variable, 304
- Subprogram, 311
- Types, 300
- Wait, 302

**Examples**

- Port association and type conversion, 23
- Port association rules, 21

## F

**Fields**

- Std\_Logic\_Vector, 44
- Testbench
  - disk\element\cntlrb.vhd, 49

**File**

- Accessing non-consecutive elements, 113
- Array
  - disk\subprgm\l\_array87.vhd, 116
- Array emulation, 114

- Binary
  - Type restrictions, 284
- Declaration
  - in Procedures, 111, 113
- Multiple access
  - disk\subprgm\line87.vhd, 114
- Simulation speed, 280
  - Binary, 281
- Write
  - multiple processes, 129

**Finite state machine**

- Mealy/Moore machines, 232
- Synthesis
  - Styles, 232

**Flip-Flop**

- Synthesis
  - Asynchronous reset, 191

**Formal**

- Verification, 252

**Function**

- Conversion
  - Std\_Logic\_Vector to Enumeration, 46
- Parameter types, 110
- Port map, 267
- Std\_Match
  - Numeric\_Std package, 211
- Synthesis
  - Latch Inference, 193
- To\_Integer
  - Numeric\_Std package, 209

**Functional**

- Verification, 247

## G

**Garbage Collection**

- Access type, 106

**Generate**

- Application example, 162
- Configuration Declaration, 33
- reusability, 328
- Reuse, 328
- Statement
  - Salient points, 9

**Generics**

- Applications, 319
- Static, 319

**Global**

- Signal, 285

**Globally static,**

- Definition, 318
- Restrictions, 319

**Gound**

- Port mapping, 239

**Guard**

- Synthesis, 185
- Signals, 11

**Guidelines**

- Buffer port, 17
- Component declaration, 25
- Configuration declaration, 25, 29
- Constants, 279
- Deallocation
  - in Subprograms, 107
- Don't care, 212
- Enhancing simulation speed, 270
- Enumerated type, 279
- File alternatives, 280
- Files
  - Use of, 280
- Memories, 279
- Processes, 278
- Resolved signals, 272
- Signal
  - Utilization, 271
- Synthesis
  - Asynchronous reset, 192
  - Don't care, 212
  - Tri-State, 229
  - Variable
    - Initialization, 193
- Type conversion, 279

## H

**Hexadecimal**

- Conversion, 122

**Hierarchy**

- Signal
  - Visibility, 289

**Homograph, 64**

- declarations, 64

## I

**Identifier**

Declaration reference, 67

**IEEE**

Standards, 143  
Standards VHDL, 143  
Web site, 143

**If**

Priority, 210  
Synthesis, 210

**Image Package, 122**

Application, 126  
Example, 61

**Initialization**

Synthesis, 184

**InOut Port, 20**

Buffer Port, 17

**Integer**

Range, 38  
Resolution function, 51, 54  
Two's complement, 38

**Internet**

FAQ, 294

## L

**Latch**

Inference, 188

**Leak**

Memory  
Deallocate, 106  
ReadLine, 108

**LFSR (Linear Feedback Shift register)**

Application example, 257  
Example, 51  
Synthesis, 132, 136  
Verification, 249

**Line (TextIO)**

Signal, 291

**Linked List**

Example, 154

**Locally static, 318****Loop Statement**

Synthesis, 210

**LPM**

Reuse, 329

**LRM, 64**

*1.1.1.2 -- Ports, 20*  
12.4.3  
Configuration  
Declaration  
Deferred Binding, 31  
12.6.1 Drivers, 83  
14.1 – Predefined attributes, 302  
3.2.1 Arra types, 71  
4.3.1.2 Drivers, 84  
4.3.2.2  
Subelement association, 230  
6.5  
Null Slice, 66  
7.2.2  
Equality, 63  
7.3.1, 44  
7.3.2.2, 76  
7.3.5, Type conversion, 39  
7.4 – Static expression, 70  
7.4.2, Static Expression, 92  
8.4.1, Side effects (Procedure), 100

## M

**Memory**

Leak  
Deallocate, 106  
ReadLine, 108  
Model, 146  
Simulation Speed, 284

**Metalogical**

Definition, 149

**MISR (Multiple Input Signature Register)**

Data reduction, 281  
instead of file, 281  
Regression  
Verification, 251



**Mod, 40**

disk\element\count\_ea.vhd, 41

**Model**

Array, timing specification, 57

Association VCC/GND

disk\synths\togndvcc.vhd, 239

Barrel shifter, 205

Behavioral

disk\synths\barrelnv.vhd, 205

Synthesis

disk\synths\barlsyn1.vhd, 206

disk\synths\barrel.vhd, 207

Binary File

Read

disk\potpourri\lntb93.vhd, 283

Write

disk\potpourri\lnta93.vhd, 282

Bit Reversal function

disk\synths\bitrevs.vhd, 240

Bit reversal process

disk\synths\bitrevp.vhd, 240

Bit Reverse

disk\subprgm\bitrfunct.vhd, 119

Constants

Application of

disk\potpourri\const.vhd, 280

Conversion

Integer to time

disk\subprgm\timecnvt.vhd, 117

Copy file VHDL'87, 112

Copy file VHDL'93, 113

Core

Designs, 329

Count-Down

disk

element\count\_ea.vhd, 41

Counter, 287

Deferring binding, 31

Demultiplexer, 203

disk\synths\demuxc.vhd, 204

disk\synths\demuxp2.vhd, 205

disk\synths\demuxpl.vhd, 204

Error injector, 163

disk\models\errinj.vhd, 170

Error injector configurations

disk\models\ej\_cfg.vhd, 177

Error injector testbench

disk\models\ej\_tb.vhd, 176

Flip-flop

Asynchronous reset:, 191

disk\synths\dff.vhd, 196

Global signal

disk\potpourri\hierglob.vhd, 287

**Integer**

Random package

disk\verifc\uniform.vhd, 260

Linear Feedback Shift Register

disk\package\lfsr4.vhd, 135

MC14532B

Cacade, 215

Priority encoder, 213

**Memory**

C language, 157

Configuration

disk\ramfix\_c.vhd, 158

disk\ramlnk\_c.vhd, 159

disk\rampag\_c.vhd, 158

Disk paging, 156

Dynamic page, 154

disk\models\ramlink.vhd, 156

Fixed Array, 149

disk\models\ramfix.vhd, 152

Fixed page, 152

disk\models\rampage.vhd, 154

Testbench, 157

disk\models\ram\_tb.vhd, 158

Traditional, 146

disk\models\memory.vhd, 149

Multiplexer, 202

disk\synths\mux.vhd, 203

Mutiplier

disk\synths\multpy.vhd, 243

Pre-charge, 220

Priority Encoder, 212

Register file

disk\synths\regfile.vhd, 200

Variables, 200

Shared variable

TextIO.Line

disk\potpourri\lnta93.vhd, 292

Shift register, 197

disk\synth\regok.vhd, 198

disk\synths\reg9.vhd, 197

disk\synths\regok2.vhd, 199

Strings to screen

disk\models\strgnout.vhd, 128

Synthesis

Counter

disk\synths\count.vhd, 189

Counter latch

disk\synths\countvar.vhd, 190

Flip-Flop

synths\d2ff\_asyn.vhd, 192

synths\dffasyn.vhd, 191

synths\dffsync.vhd, 191

TextIO.Line

String

disk\potpourri\lnta87.vhd, 293

**Timer**

Integer

\disk\synths\timer\_ea.vhd, 241

Unsigned

\disk\synths\timr3\_ea.vhd, 242

**Waveform generation**

disk\element\wave.vhd, 8

**Zero Ohm bank**

disk\models\zohm\_ea.vhd, 162

**Zero Ohm resistor, 159**

disk\models\z0quiet.vhd, 161

disk\models\zohm0\_ea.vhd, 160

**Modeling**

Behavioral/RTL, 296

**Models for Core**

Reuse, 329

**Multiple Input Signature Register**

see MISR

**Multiplexer**

Synthesis, 202

**Multiplier**

Synthesis:, 242

**N****Non-Associative Operator, 41**

disk\element\left2r.vhd, 70

**Non-Portable**

Deferred Binding, 31

**Normalization**

Array

in Subprogram, 118

**Notation**

Recommended, 316

**Null**

Array, 65

Synthesis

Example, 206

**Numeric\_Bit**

Source, 143

**Numeric\_Std**

Source, 143

**Numeric\_Std package**

Signed/Unsigned, 60

**O****One-Hot encoding**

Synthesis, 233

**Operator**

&amp;

in Case statement, 42

Long strings, 43

Implicit

Array, 58

Non-Associative, 41

disk\element\eft2r.vhd, 70

Overloading

Equal, 63

Synthesis (in), 184

**Others**

See Array, 76

Synthesis, 76

aggregates, 75

**Out Port, 19**

disk\element\buffer.vhd, 20

**Overload**

Operator

Explicit, 61

Equal, 63

Exam, 306

**P****Package**

Accelerated - list, 285

Conversion

Integer to time, 116

typed objects to string, 122

disk\package\image\_pb.vhd, 126

Image

disk\package\image\_pb.vhd, 126

Linear Feedback Shift Register, 132

disk\package\lfsrstd.vhd, 133

Multiple Input Shift Register, 130

Multiple Input Signature Register

disk\package\misr\_pb.vhd, 131

Numeric\_Std

Signed/Unsigned, 60

To\_Integer function, 209

- Random generator
    - disk\package\randm\_int.vhd, 137
    - Integer, 137
  - Std\_Logic\_Arith
    - Signed /Unsigned, 60
  - Synthesis
    - Supported, 184
  - Parameterization**
    - Accordion, 318
    - Alias, 320
    - Control, 319
    - Records, 320
    - Subtypes, 320
  - Parameterized**
    - Priority Encoder, 212
  - Parameters**
    - Accordion/control, 316
  - Path definition, 317**
  - Pipeline**
    - Synthesis, 224
  - Port**
    - Association list
      - disk\element\alist\_ea.vhd, 23
      - Type conversion, 22
    - Association rules
      - Open, 20
    - InOut, 20
    - InOut versus Buffer, 17
    - Out, 19
    - Source, 93
    - Synthesis
      - Initialization, 184
  - Port Association**
    - Open
      - disk\element\open\_ea.vhd, 22
  - Port map**
    - Function call (in), 267
  - Printing**
    - from VHDL, 126
  - Priority**
    - Case statement, 209
    - IF Statement, 210
  - Priority Encoder, 212**
  - Procedure**
    - Concurrent, 193
    - File declaration, 111
    - Scope of visibility, 101
    - Side Effects, 100
  - Process**
    - Clocked
      - Synthesis:, 186
    - disk\element\process.vhd, 7
    - Salient points, 6
    - Sensitivity
      - Activity, 89
      - Resolved signal, 87
      - Std\_Logic\_Vector, 87
    - Variable
      - Lifetime, 193
  - Pre-charge**
    - Model, 220
- Q**
- Qualifier (type), 44**
- R**
- Random**
    - Package, 137
    - Integer, 137
    - LFSR, 51
  - Range constraint, 50**
  - Range definition**
    - Methods, 303
  - Record**
    - Parameterization, 320
  - Reference**
    - [10], 131
    - [11], 183
    - [12], 195
    - [13], 195
    - [14], 235
    - [15], 252
    - [16], 270
    - [17], 220
    - [2], xxvii, 245, 270
    - [3], xxvii
    - [4], xxvii

- [5], xxviii
- [6], 44, 48, 273
- [7], 58
- [8], 92
- [8], 245
- [9], 104

**Register**

- Shadow
- Synthesis, 224

**Register file**

- Signal, 199
- Variable, 200

**Register model**

- Synthesis, 186

**Regression**

- MISR/LFSR example, 257
- Verification, 250

**Rem, 40****Requirements**

- in Design Processes, 314

**Resolution Function**

- Boolean
- Exam, 309
- Array, 86

**Resolved signal**

- Activity, 85

**Resource sharing**

- Synthesis, 235

**Reuse**

- Attributes, 302, 328
- BFM, 330
- Block, 323
- Coding styles, 316
- Core
- LPM, 329
- Core models, 329
- Design for, 313
- Design Processes, 314
- DesignWare, 329
- Documentation, 331
- Generate, 328
- Parameters, 316
- Scaleability, 334
- Subprogram, 328

**Rollover**

- Methods, 241

**S****Scaleability**

- Reuse, 334

**Selected Signal Assignment**

- Example, 7

**Semaphore**

- Shared variable, 50

**Sensitivity**

- Resolved composite, 87

**Sensitivity rule**

- Synthesis, 186

**Sequential statement**

- Synthesis, 185

**Shared variable, 50, 291**

- Example
- Line, 291
- Semaphore, 50

**Shift Register**

- Synthesis, 197

**Side effects**

- Avoidance method
- Client/server, 103
- Concurrent procedure, 8
- in Concurrent procedure, 8
- in Procedure, 100

**Signal**

- Access type
- TextIO.Line, 291
- Exam, 304
- Global, 285
- Hierarchy, 285
- Hierarchy
- Visibility, 285
- Internal to component, 285
- Simulation speed
- Composite, 271
- Implicit, 272
- Reassignment, 271
- Resolved, 272
- Synthesis
- Bus, 184

- Initialization, 184
  - Register, 184
  - Utilization, 271
- Signed**
- Operations, 60
- Simulation**
- Stopping, 281
- Simulation speed**
- Composite assignment, 271
  - Concurrent statement
    - Disabling, 275
  - Concurrent signal assignment, 273
  - Concurrent statement, 273
  - Constant, 279
  - Enhancing methods, 270
  - File, Binary, 280, 281
  - Generate, 276
  - MISR, 281
  - Processes, 273
  - Redundant code, 277
  - Signal
    - Implicit, 272
    - Resolved, 272
  - Signal, 270
  - Signal reassignment, 271
  - Subprograms, 284
  - Types
    - Integer/Std\_Logic, 278
  - Verifier, 281
  - VITAL, 285
- Slices**
- Null
  - Synthesis, 65
- Standards VHDL**
- IEEE, 143
- Static**
- Global/Local, 318
- Static expression**
- LRM
    - 7.4.2, 92
    - 7.4, 70
  - range constraint, 50
- Std\_Logic**
- VHDL'87/VHDL'93
    - Differences, 43
    - disk\element\confused.vhd, 44
- Std\_Logic\_Arith package**
- Signed/Unsigned, 60
- Std\_Logic\_Vector**
- Resolution, 162
- Stopping**
- Simulation, 281
- Strength stripper, 267**
- String**
- Long, 43
  - See conversion, 122
- Subelement Association**
- Application example, 173
  - in ports, 231
  - LRM 4.3.2.2.2, 230
  - Synthesis, 230
- Subprogram**
- Exam, 311
  - Normalization of arrays, 118
  - Others, 76
  - overloading
    - LRM 2.3, 64
  - reusability, 328
  - Reuse, 328
  - See Procedure/Function, 100
  - Simulation Speed:, 284
  - Variable
    - Initialization, 195
- Subtype**
- Range definition, 303
  - Exam, 303
  - Integer
    - Parameterization, 320
- Synthesis, 65**
- Aggregate, 185
  - Alias, 184
  - Also see model, 186
  - Architecture, 184
  - Array, 73, 186
    - Allowed, 56
  - Array
    - Memory, 57
    - Two-dimensional, 57, 199
  - Assertion, 185
  - Assignment
    - Signal, 185

- Barrel shifter, 205
- Behavioral, 243
- Bit Reversal, 240
- Block
  - Guarded, 17
  - Restrictions, 185
- Block Statement, 10
  - Guarded, 17
- Buffer drivers, 225
- Bus, 184
- Case
  - & operator, 42
  - Select, 70
- Component
  - Predefined, 199
- Concurrent statements, 185
- Configurations, 185
- Constructs
  - Supported/Unsupported, 184
- Count-Down
  - disk
    - element/count\_ea.vhd, 41
- Demultiplexer, 203
- DesignWare, 234
- Documentation, 336
- Don't care, 207
- EDIF, 336
- Entity, 184
- Enumeration
  - Enum\_Encoding, 47
- File, 184
- Finite State Machine
  - Styles, 232
- Flip-flop
  - Asynchronous reset, 191
- Function
  - Latch inference, 193
- Generate statement, 9
- Guard, 185
- Guarded signals, 12
- If statement, 210
- Inequality operator, 238
- Initialization
  - Object, 184
- Integer subtype, 208
- Latch
  - Inference, 188
- Latch/register, 186
- Latches*
  - in Tri-states*, 227
- LFSR register, 136
- Loop statement, 210
- Memory, 199
- Model documentation, 336
- Multiplexer, 202
- Multipplier, 242
- Now, 185
- Null array
  - Example, 206
- One-Hot encoding, 233
- Operators, 184
- Optimizing for area, 238
  - Design, 236
- Package
  - Supported/Unsupported, 184
- Pipeline, 224
- Port Mapping to VCC/Ground, 239
- Register
  - Inference, 188
- Register File, 199
- Register outputs
  - Need for, 225
- Register/Combinational, 188
- Resolution function
  - User defined, 185
- Resource sharing, 235
  - Styles, 235
- Sensitivity rule, 186
- Sequential statement, 185
- Shadow register, 224
- Shift Register, 197
- Signal
  - Register, 184
- Simulation mismatch, 83
- Slice
  - Computational, 206
  - Others, 76
- Std\_Logic\_Arith
  - + integer, 189
- Subelement association, 230
- Technology impact, 223
- Timer
  - Up/Down
    - Techniques, 241
- Tri-State
  - Guideline, 229
  - Rules, 226
- Type
  - Supported, 184
- Types, 95
- Unresolved
  - Arithmetic operations, 95
- Unresolved type
  - usage, 95
- Variable, 186, 193, 197
  - Initialization, 193
    - Process, 193
    - Register, 197
- Variable Initialization
  - Subprogram, 193, 195
- Wait, 185
  - Restrictions, 195

## T

**Technology**

- Synthesis, 223

**Testbench**

- BFM

- Design, 254
- Documentation, 338
- MISR/LFSR example, 258
- Validation Plan, 248

**TextIO**

- Std\_Logic
- Application, 122

**Timer**

- Up/Down
- Techniques, 241

**Timing checkers**

- Verification, 249

**Tri-State**

- Synthesis, 226

**Two's complement**

- Integer, 38

**Type**

- Array (LRM 3.2.1), 71
- Exam, 300
- Conversion, 39
  - disk\element\convn.vhd, 40
- Qualifier, 39

**Type conversion**

- Port association lists, 22

## U

**Unconstrained Array Aggregate, 75****Unsigned**

- Operations, 60

## V

**Variable**

- Exam, 304
- Initialization
  - Subprogram, 195
  - Synthesis, 193
- Lifetime, 193
- Process, 193

- Shared, 50, 291

- Synthesis

- Initialization, 184, 193

**VCC**

- Port mapping, 239

**Verification**

- Automatic, 247
- Design, 248
- Design verifier
  - Technique, 251
- Documentation, 337
- Error injection, 163
- File compare
  - Technique, 250
- Formal, 252
  - Regression, 252
- Functional, 247
- LFSR
  - Application example, 257
  - Technique, 249
- MISR
  - Application example, 257
- Processes, 246
- Pseudo-random
  - Data, 249
  - Transaction, 249
- Regression, 246
  - MISR, 251
  - Techniques, 250
- Timing checkers, 249

**Verifier**

- Data reduction, 281

**VHDL**

- '98
  - Requests, 272
  - Item for review, 76
- VHDL'87/93 Difference, 43

**Visibility**

- Scope, 101

**VITAL**

- Packages:, 295
- Purpose, 295
- Simulation Efficiency, 285
- Simulation Speed:, 285
- Source, 143

## W

### **Wait**

- Exam, 302

- Simulator handling, 273

- Synthesis

  - Restrictions, 195

- Synthesis:, 185

### **Waveform**

- Generation example, 8

### **Waveform generation**

- disk\element\wave.vhd, 8