WL-TR-91-5030

VERY HIGH SPEED INTEGRATED CIRCUITS (VHSIC) HARDWARE DESCRIPTION LANGUAGE (VHDL) SYNTAX AND SEMANTICS SUMMARY

Michael T. Mills
Design Branch
Microelectronics Division

January 15, 1991

Final Report for Period June 1990 to June 1991

Approved for public release; distribution unlimited.

SOLID STATE ELECTRONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6543

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Michael T. Mills

MICHAEL T. MILLS, LtCol, USAFR
Design Branch
Microelectronics Division

John W. Hines

JOHN W. HINES, Chief
Design Branch
Microelectronics Division

FOR THE COMMANDER

Stanley E. Wagner

STANLEY E. WAGNER, Chief
Microelectronics Division
Solid State Electronics Directorate

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WL/ELED , *W-PAFB, OH 45433 to help us maintain a current mailing list.*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	15 January 1991	FINAL June 1990 - June 1991

4. TITLE AND SUBTITLE
Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL) Syntax and Semantics Summary

5. FUNDING NUMBERS
PE 62204F
PR 6096
TA 40
WU 18

6. AUTHOR(S)
Mills, Michael T.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Darrell Barker, (513) 255-4448
Solid State Electronics Directorate (WL/ELED)
Wright-Patterson AFB OH 45433-6543

8. PERFORMING ORGANIZATION REPORT NUMBER
WL-TR-91-5030

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The computer code contained herein are theoretical and/or references that in no way reflect Air Force-owned or -developed computer software.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This is a depth-first ordered VHDL syntax summary with annotated semantics and corresponding paragraph numbers to VHDL and Ada Reference Manuals. The BNF representation of VHDL is cross-referrenced to comparable VHDL and Ada syntax and semantic descriptions within the VHDL and Ada language reference manuals (LRM).

14. SUBJECT TERMS
VHDL-IEEE 1076-Design Language-Hardware Description Language Ada

15. NUMBER OF PAGES
41

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

I. Introduction

This product resulted from the first step of developing a VHDL to Ada interface. This ordering of the BNF representation of VHDL should help reduce the time for new VHDL implementors and programmers to overcome the robustness of the VHDL syntax and semantics. It could serve as an implementor's guide for developing VHDL software tools and help VHDL programmers remember the details of the VHDL syntax and semantics without having to thumb through the scattered bits and pieces of syntax scattered in the syntax summary of the reference manual. This VHDL Summary could also help insure validation test coverage by aiding the validation test writer to keep track of each detail incorporated in the VHDL language. The annotated semantics consist of brief descriptions to serve as reminders. It is not meant to take the place of the detailed text in the VHDL reference manual. Also, the behavioral portions of the VHDL syntax include commented Ada Reference Manual paragraph numbers when corresponding Ada functionality exists. All VHDL syntax and semantics included in this product are taken from the VHDL reference manual but are reordered for clarity. The semantics in this summary is not completely defined, but includes enough detail to guide the programmer or implementor.

```
        Note: "<=", "=>", and "<>" are terminals and should be distinguished
        from the non-terminal <text>.
        Non-terminals are represented by enclosing "<" and ">".
        Italic portions of non-terminals in the reference manual
        are designated by enclosing them in quotes <"text"_more_text>.
        All comments are preceded by double dashes -- .
        Everything except VHDL syntax is in comments, preceded by -- .
        Therefore, the reordered BNF description of the VHDL syntax can
        be used to drive a VHDL parser for automated tools.
        VHDL paragraph numbers are preceded by "-- VHDL".  When they
        exist, corresponding Ada paragraph numbers are preceded by
        "-- Ada".  (Where feasible, parts of productions with
        corresponding Ada equivalence are followed by "--a".
        Note: <underline> is equivalent to the terminal "_".
        The symbol "^" refers to the line above the current one.
        Semantic annotations (comments) are preceded by "-- SEM:".
        Most up-level references are symbolized by "----^".

--:*1:
<design_file> ::= <design_unit> { <design_unit> } --:2,2: -- VHDL 11.1

        -- SEM: Design units may be independently analyzed and inserted into
        --      design library.  They are analyzed in textual order.
        --      A design library is an implementation dependent storage of
        --      previously analyzed design units.


    <design_unit> ::= <context_clause> <library_unit>     -- VHDL 11.1
                                                  --:3,8:

--:*3:
        <context_clause> ::= <context_item>       --:4:  -- VHDL 11.3

--:*4:
            <context_item> ::= <library_clause> | <use_clause>  -- VHDL 11.3
                                                  --:5,180:
--:*5:
                <library_clause> ::= library <logical_name_list> ;  -- VHDL 11.2
                                                  --:6:

                        -- SEM: This defines logical names for design libraries.
                        --      Scope extends to end of declarative region
                        --      associated with design unit.
                        --      Each library name denotes a design library.
                        --      Two classes of design libraries: (See 11.2)
                        --         Working libraries and resource libraries.

--:*6:
                    <logical_name_list> ::= <logical_name> { , <logical_name> }
                                                  --:6,6:         -- VHDL 11.2
```

```
--:*7:
                    <logical_name> ::= <identifier>      --:49: -- VHDL 11.2

                        -- SEM: STD denotes library where package STANDARD
                        --      and package TEXTIO reside. (See Chapt 14).

            -- <use_clause> ::=   --:180:   (see below) -- VHDL 11.2 Ada 8.4

                    -- SEM: This makes declarations visible in design unit
                    --      and defines dependencies among design units.

--:*8:
        <library_unit> ::= <primary_unit> | <secondary_unit>
                                          --:9,10:     -- VHDL 11.1 Ada 10.1
          -- SEM: Secondary unit is a separately analyzed body of primary unit.

--:*9:
            <primary_unit> ::=                          -- VHDL 11.1 Ada 10.1
                <entity_declaration>                       --:11:
                | <configuration_declaration>              --:56:
                | <package_declaration>                    --:63:

--:*10:
            <secondary_unit> ::=                        -- VHDL 11.1 Ada 10.1
                <architecture_body>                        --:66:
                | <package_body>                           --:71:

                -- SEM: Of secondary units, only architecture bodies are named.
                --      Primary units within library and architecture bodies
                --      associated with a given entity must have unique names.
                --      Must analyze all primary units before analyzing design
                --      unit which uses (or references) them and before
                --      corresponding secondary units.

        -- Note: A design entity is the primary hardware abstraction in VHDL.
        --       A collection of design entities form a design hierarchy.  An
        --       <entity declaration> can potentially represent a "class" of
        --       design entities, each with the same "interface".
        --       <configuration_declaration> describes how design entities are
        --       put together.


  ------------------------------------------------------------------------


--:*11:
   <entity_declaration> ::=                             -- VHDL 1.1
    entity <identifier> is                  --:49:
      <entity_header>                       --:12:
```

```
      <entity_declarative_part>                --:52:
    [ begin <entity_statement_part> ]          --:54:
    end [ <"entity"_simple_name>               --:178:

--:*12:
       <entity_header> ::=                                  -- VHDL 1.1.1
         [ <"formal"_generic_clause> ]         --:13:
         [ <"formal"_port_clause> ]            --:14:

              -- SEM: declares "objects" for communicating with environment.

--:*13:
          <generic_clause> ::=                              -- VHDL 1.1.1
            generic ( <generic_list> );    -- Note: Completely different
                                  --:15:  --       from Ada generic.
              -- SEM: values (constants) may be determined by environment
              --      via <"generic"_association_list> of a component instant.
              --      Generics may control structural, dataflow, or behavioral
              --      characteristics of block or simply document.
              -- SEM: ERROR if no actual or no default is specified.

--:*14:
          <port_clause> ::=                                 -- VHDL 1.1.1
            port ( <port_list> );    --:16:

              -- SEM: defines I/O ports of design entity.

              -- SEM: Both can be visible outside of design entity.

--:*15:
                    <generic_list> ::= <"generic"_interface_list>
                                          --:17:    -- VHDL 1.1.1.1
                        -- SEM: defines generics of a block.  Each interface
                        --      element declares a formal generic.

--:*16:
                    <port_list> ::= <"port"_interface_list>
                                          --:17:    -- VHDL 1.1.1.2
                        -- SEM: defines communication channels between
                        --      block and environment.  Ports are signals.
                        --      The actual (or calling) parameter must be
                        --      a static name.
                        --      See 1.1.1.2 for list of restrictions.

--:*17:
                      <interface_list> ::=
                        <interface_element> { <interface_element> }
                                          --:18,18:  -- VHDL 4.3.3.1

                           -- SEM: "Generic" interface list consist
                           --       entirely of interface "constant"
```

```
                                --       declarations.
                                --       "Port" interface list consists
                                --       entirely of interface "signal"
                                --       declarations.
                                --       "Parameter" interface list may contain
                                --       interface "constant", "signal", or
                                --       "variable" declarations.
                                --       These are interface "objects" required
                                --       by a subprogram, component, design
                                --       entity, or block statement.
                                --       (See pages 2-2 thru 2-4 on parameters.)

--:*18:
                          <interface_element> ::= <interface_declaration>
                                              --:19:       -- VHDL 4.3.3.1

--:*19:
                             <interface_declaration> ::=  -- VHDL 4.3.3
                                  <interface_constant_declaration>  --:20:
                                | <interface_signal_declaration>    --:21:
                                | <interface_variable_declaration>  --:22:
                                  --
      -- -- -- -- -- -- -- -- -- -- -- --
     --
--:*20:
     <interface_constant_declaration> ::=                  -- VHDL 4.3.3
          [constant] <identifier_list> : [in] <subtype_indication>  --:17,23:
                                       [:= <"static"_expression> ]    --:27:

             -- SEM: Subtype cannot be file type or access type.

     --
--:*21:
     <interface_signal_declaration> ::=                    -- VHDL 4.3.3
          [signal] <identifier_list> : [mode] <subtype_indication> [bus]
                                          [:= <"static"_expression> ]
                                               --:17,23,27:

             -- SEM: Subtype cannot be file type or access type.
             --      "Bus" means signal is guarded and of signal kind bus.
             --      See pg. 4-9 and 4-10 for more detailed restrictions.

     --
--:*22:
     <interface_variable_declaration> ::=                  -- VHDL 4.3.3
          [variable] <identifier_list> : [mode] <subtype_indication>  --:17,23:
                                          [:= <"static"_expression> ]  --:27:
                                             --
               --------------------------------
--:*23:           --
               <subtype_indication> ::=                    -- VHDL 4.2 Ada 3.3.2
```

```
                [ <"resolution_function"_name> ] <type_mark> [<constraint>]
                                                              --:34, 4,25:

                    -- SEM: Resolution function means signal of that
                    --      subtype is resolved by named function
                    --      (see 2.4).  It defines how values of multiple
                    --      sources of a signal are resolved into a
                    --      single value.
                    --      Subtype indication denoting access type or
                    --      file type cannot contain resolution function.
                    --      (See both 2.4, 4.2 and 4.3.1.2 on resolution
                    --      function.)

--:*24:
                  <type_mark> ::= <"type"_name> | <"subtype"_name>   --::
                                        --:34,34:  -- VHDL 4.2 Ada 3.3.2
--:*25:
                  <constraint> ::= <range_constraint> | <index_constraint>
                                      --:140,159:    -- VHDL 4.2 Ada 3.3.2

--:*26:           --
              <identifier_list> ::= <identifier> [, <identifier> ]     --::
                                      --:49,49:      -- VHDL 3.2.2 Ada 3.2
  ------------------------------------------------------------------------

  --   Here are some low level definitions before we go on:

--:*27:
      <expression> ::=                                  -- VHDL 7.1   Ada 4.4
           <relation> { and <relation> }                                --a
         | <relation> { or <relation> }                                 --a
         | <relation> { xor <relation> }                                --a
         | <relation> { nand <relation> }
         | <relation> { nor <relation> }        --:28,........:
                                      -- Note: ... means repeat reference.

--:*28:
          <relation> ::=                                -- VHDL 7.1 Ada 4.4
             <simple_expression> [ <relational_operator> <simple_expression> ]
                                                            --:29,48,29:
--:*29:
             <simple_expression> ::=                    -- VHDL 7.1 Ada 4.4
                 [ <sign> ] <term> { <adding_operator> <term> }  --:30,47,31:

--:*30:
                  <sign> ::= + | -                      -- VHDL 7.2

--:*31:
                          <term> ::=                    -- VHDL 7.1 Ada 4.4
                              <factor> { <multiplying_operator> <factor> }
                                                          --:32,46,32:
```

```
--:*32:
                        <factor> ::=          -- VHDL 7.1 Ada 4.4
                            <primary> [ ** <primary> ]
                            | abs <primary>        --:33:
                            | not <primary>        --:33:

--:*33:
                             <primary> ::=    -- VHDL 7.1 Ada 4.4
                                 <name>                        --:34:
                                 | <literal>                   --:37:
                                 | <aggregate>                 --:89:
                                 | <function_call>             --:87:
                                 | <qualified_expression>      --:43:
                                 | <type_conversion>           --:44:
                                 | <allocator>                 --:45:
                                 | ( <expression> )            --:27:

--:*34:
                                 <name> ::=    -- VHDL 6.1 Ada 4.1
                                   <simple_name>               --:178:
                                   | <operator_symbol>         --:126:
                                   | <selected_name>           --:181:
                                   | <indexed_name>            --:35:
                                   | <slice_name>              --:36:
                                   | <attribute_name>          --:95:
                                        --
                    ---------------------------
--:*35:              --
                    <indexed_name> ::=            -- VHDL 6.4 Ada 4.1.1
                        <prefix> ( <expression> { , <expression> } )
                                                        --:96,27,27:

--:*36:
                    <slice_name> ::= <prefix> ( <discrete_range> )
                                                  -- VHDL 6.5 Ada 4.1
                                                        --:96,93:

--:*37:
                <literal> ::=                     -- VHDL 7.3.1 Ada (4.4)
                   <numeric_literal>                        --a --:38:
                   | <enumeration_literal>                     --:135:
                   | <string_literal>                       --a --:39:
                   | <bit_string_literal>                         --:40:
                   | null                                   --a

--:*38:
                   <numeric_literal> ::=          -- VHDL 7.3.1 Ada 2.4
                       <abstract_literal>                      --:146:
                       | <physical_literal>                    --:145:
```

```
--:*39:
                    <string_literal> ::= " { <graphic_character> } "
                                          --:137: -- VHDL 13.6  Ada 2.6

--:*40:
                    <bit_string_literal> ::=        -- VHDL 13.7
                          <base_specifier> " <bit_value> "  --:41,42:

--:*41:
                              <base_specifier> ::= B | O | X  -- VHDL 13.7

--:*42:
                              <bit_value> ::=                  -- VHDL 13.7
                                  <extended_digit>           --:153:
                                  { [ <underline> ] <extended_digit> }
                                                            --:_,153:
                                     -- Note: <underline> is a lexical
                                     -- representation of terminal "_".

--:*43:
                <qualified_expression> ::=          -- VHDL 7.3.4 Ada 4.7
                    <type_mark> ' ( <expression> )              --:24,27:
                    | <type_mark> ' <aggregate>                 --:24,89:

--:*44:
                <type_conversion> ::= <type_mark> ( <expression> )
                                       --:24,27:    -- VHDL 7.3.5 Ada 4.6

--:*45:
                <allocator> ::=                     -- VHDL 7.3.6
                     new <subtype_indication>       ----^  --:23:
                     | new <qualified_expression>   ----^  --:43:

--:*46:
            <multiplying_operator> ::= * | \ | mod | rem  -- VHDL 7.2 Ada 4.5

--:*47:
            <adding_operator> ::= + | - | &         -- VHDL 7.2

--:*48:
            <relational_operator> ::=  = | /= | < | <= | > | >=
                                                    -- VHDL 7.2 Ada 4.5

       ----------------- (end of <expression>)

--:*49:
      <identifier> ::=                              -- VHDL 13.3 Ada 2.3
          <letter> { [ <underline> ] <letter_or_digit> }   --:50,_,51:
                                  -- <underline> is a lexical element.

--:*50:
```

```
        <letter> ::= <upper_case_letter> | <lower_case_letter> -- See Note
                                                    -- VHDL 13.3 Ada 2.3

--:*51:
            <letter_or_digit> ::= <letter> | <digit>  -- VHDL 13.3 Ada 2.3
                                  --:50:   -- <digit> is a lexical element.

                -- Note: <underline>, <upper_case_letter>, and
                --       <lower_case_letter> are lexical representations of
                --       terminals.

        ---------------- (end of <identifier>)
```

```
--:*52:
      <entity_declarative_part> ::= {<entity_declarative_item>}  -- VHDL 1.1.2
                                                                 --:53:
          -- SEM: Declarations are common to all design entities whose
          --      interfaces are defined by entity declaration.

--:*53:
         <entity_declarative_item> ::=                  -- VHDL 1.1.2
             <subprogram_declaration>           --:122:
             | <subprogram_body>                --:127:
             | <type_declaration>               --:130:
             | <subtype_declaration>            --:166:
             | <constant_declaration>           --:167:
             | <signal_declaration>             --:184:
             | <file_declaration>               --:169:
             | <alias_declaration>              --:172:
             | <attribute_declaration>          --:173:
             | <attribute_specification>        --:174:
             | <disconnection_specification>    --:186:
             | <use_clause>                     --:180:


--:*54:
      <entity_statement_part> ::= { <entity_statement> }  -- VHDL 1.1.3 --:55:

--:*55:
         <entity_statement> ::= <concurrent_assertion_statement>  -- VHDL 1.1.3
                              | <"passive"_concurrent_procedure_call>
                              | <"passive"_process_statement>
                                    --:74,77,166:

             -- SEM: All such statements are passive (see 9.2).
```

```
--      They may monitor operating conditions or characteristics
--      of design entity.

-------------------------------------------------------------------------

--:*56:
<configuration_declaration> ::=                          -- VHDL 1.3
    configuration <identifier> of <"entity"_name> is     --:49,34:
       <configuration_declarative_part>                  --:57:
       <block_configuration>                             --:58:
    end [ <"configuration"_simple_name> ] ;              --:178:

          -- SEM: Specifies deferred bindings of component instances of
          --      design entities.  Must reside is same library as it
          --      corresponding entity declaration.  <entity_name> is
          --      apex of design hierarchy.  Elaboration achieves effect.
          --      Used to build more complex configurations.

--:*57:
       <configuration_declarative_part> ::=              -- VHDL 1.3
            { <configuration_declarative_item> }            --:57a:

--:*57a:
             <configuration_declarative_item> ::=        -- VHDL 1.3
                 <use_clause>                               --:180:
                 | <attribute_specification>                --:174:

--:*58:
        <block_configuration> ::=                        -- VHDL 1.3.1
            for <block_specification>                       --:59
                { <use_clause> }                            --:180:
                { <configuration_item> }                    --:61:
            end for;

--:*59:
               <block_specification> ::=                 -- VHDL 1.3.1
                   <"architecture"_name>                    --:34:
                   | <"block_statement"_label>              --:75:
                   | <"generate_statement"_label> [ ( <index_specification> ) ]
                                                            --:75,60:

                       -- SEM: identifies internal or external block to which
                       --      block configuration applies.
                       --      See 1.3.1 for detailed restrictions.

--:*60:
                     <index_specification> ::=           -- VHDL 1.3.1
                         <discrete_range>                   --:93:
                         | <"static"_expression>           --:27:

--:*61:
```

```
                    <configuration_item> ::=                -- VHDL 1.3.1
                         <block_configuration>                  --:58:
                         | <component_configuration>            --:62:

--:*62:
                           <component_configuration ::=     -- VHDL 1.3.2
                               for <component_specification>        --:191:
                                  [ use <binding_indication> ; ]    --:193:
                                  [ <block_configuration> ]         --:58:
                               end for ;

                                  -- SEM: defines configuration of one or more
                                  --      component instances in block.

                              -- SEM: applies to component instances within block.

                              -- SEM: ERROR if explicit configuration specification
                              --      (in architecture body) and component
                              --      configuration containing binding indication
                              --      apply to same component instance.
                              --      (More details on pages 1-12 and 1-13.)

                                   -- Note: <block_configuration> is recursive.

  ------------------------------------------------------------------------

--:*63:
<package_declaration> ::=                                   -- VHDL 2.5 Ada 7.1
    package <identifier> is                                 --:49:      --a
       <package_declarative_part>                           --:64:
    end [ <"package"_simple_name> ]                         --:178:     --a

          -- SEM: A subprogram written in another language can be made
          --      available by defining its interface via a subprogram
          --      declaration within a package declaration with no body.
          --      Built-in functions of simulator can use this.

--:*64:
       <package_declarative_part> ::=                       -- VHDL 2.6 Ada 7.1
           { <package_declarative_item> }                   --:65:      --a

--:*65:
             <package_declarative_item> ::=                 -- VHDL 2.5
                  <subprogram_declaration>                  --:122:     --a
                  | <type_declaration>                      --:130:
                  | <subtype_declaration>                   --:166:
                  | <constant_declaration>                  --:167:
                  | <signal_declaration>                    --:184:
                  | <file_declaration>                      --:169:
                  | <alias_declaration>                     --:172:
                  | <component_declaration>                 --:189:
```

```
                | <attribute_declaration>              --:173:
                | <attribute_specification>            --:174:
                | <disconnection_specification>       --:186:
                | <use_clause>                         --:180:     --a
```

```
--:*66:
<architecture_body> ::=                                -- VHDL 1.2
    architecture <identifier> of <"entity"_name> is    --:34:
       <architecture_declarative_part>                 --:67:
    begin
       <architecture_statement-part>                   --:69:
    end [ <"architecture"_simple_name>;                --:178:

          -- SEM: defines body of design entity.  It specifies relationships
          --      between inputs and outputs of design entity, and may express
          --      structure, dataflow, or behavior.  It must reside in same
          --      design library as its associated entity declaration.  Can
          --      have more than one architecture body per entity.

--:*67:
       <architecture_declarative_part> ::= { <block_declarative_item> }
                                                       -- VHDL 1.2.1  --:68:
              -- SEM: Declarations are available to block defined by design
              --      entity.

--:*68:
           <block_declarative_item> ::=                -- VHDL 1.2.1
                <subprogram_declaration>               --:122:
                | <subprogram_body>                    --:127:
                | <type_declaration>                   --:130:
                | <subtype_declaration>                --:166:
                | <constant_declaration>               --:167:
                | <signal_declaration>                 --:184:
                | <file_declaration>                   --:169:
                | <alias_declaration>                  --:172:
                | <component_declaration>              --:189:
                | <attribute_declaration>              --:173:
                | <attribute_specification>            --:174:
                | <configuration_specification>        --:190:
                | <disconnection_specification>        --:186:
                | <use_clause>                         --:180:

--:*69:
       <architecture_statement_part> ::= { <concurrent_statement> }  --:70:
                                                       -- VHDL 1.2.2
              -- SEM: These statements describe internal organization and/or
              --      operation of block defined by design entity.
```

```
--:*70:
            <concurrent_statement> ::=                      -- VHDL 9
                 <block_statement>                                --:112:
                 | <process_statement>                           --:166:
                 | <concurrent_procedure_call>                   --:77:
                 | <concurrent_assertion_statement>              --:74:
                 | <concurrent_signal_assignment_statement>      --:86:
                 | <component_instantiation_statement>          --:106:
                 | <generate_statement>                         --:109:

                      -- SEM: define interconnected blocks and processes that
                      --      jointly describe the overall behavior or
                      --      structure of a design.  Concurrent statements
                      --      execute asynchronously with respect to each
                      --      other.

                     --  Note: <generate_statement> includes the nonterminal
                     --        <concurrent_statement> which is recursive.

  ------------------------------------------------------------------------

--:*71:
<package_body> ::=                                          -- VHDL 2.6 Ada 7.1
    package body <"package"_simple_name> is          --:178:
       <package_body_declarative_part>               --:72:
    end [ <"package"_simple_name> ] ;                --:178:

        -- SEM: Package body is unnecessary if no subprograms or deferred
        --      constants are declared in package declaration.

--:*72:
       <package_body_declarative_part> ::=                  -- VHDL 2.6 Ada 7.1
           { <package_body_declarative_item> }              --:73:

--:*73:
            <package_body_declarative_item> ::=             -- VHDL 2.6
                <subprogram_declaration>                    --:122:     --a
                | <subprogram_body>                         --:127:
                | <type_declaration>                        --:130:
                | <subtype_declaration>                     --:166:
                | <constant_declaration>                    --:167:
                | <file_declaration>                        --:169:
                | <alias_declaration>                       --:172:
                | <use_clause>                              --:180:     --a


  ------------------------------------------------------------------------
  -- Now, continue breaking down architecture body definition:

  -- Note: The following make up <concurrent_statement>.
```

```
--:*74:
            <concurrent_assertion_statment> ::=        -- VHDL 9.4
                 [ <label> :] <assertion_statement>    --:75,75a:

                     -- SEM: represents a passive process statement
                     --      containing the specified assertion statement.
                     --      For every concurrent assertion statement, there
                     --      is an equivalent process (assertion) statement.

--:*75:
                   <label> ::= <identifier>   --:49:     -- VHDL 9.7 Ada 5.1

--:*75a:
                  <assertion_statement> ::=             -- VHDL 8.2
                      assert <condition>                --:76:
                         [report <expression> ]         --:27:
                         [severity <expression> ]       --:27:

--:*76:
                             <condition> ::= <"boolean"_expression>  --:27:
                                                        -- VHDL 8.1 Ada 5.3

  -----------------

--:*77:
            <concurrent_procedure_call> ::=            -- VHDL 9.3
                 [ <label> :] <procedure_call_statement>    --:75,78:

                           -- SEM: represents a process containing the
                           --      corresponding sequential procedure call.
                           --      For every concurrent procedure call, there
                           --      is an equivalent process statement.

--:*78:
                        <procedure_call_statement> ::=   -- VHDL 8.5 Ada 6.4
                            <"procedure"_name> [( <actual_parameter_part> )]
                                                   --:34,79:

                                  -- SEM: invokes the execution of a
                                  --      procedure body.
                                  --      Execution of a procedure call
                                  --      includes evaluating actual
                                  --      parameters and expressions
                                  --      associated with formal parameters.

--:*79:
                                <actual_parameter_part> ::=
                                    <"parameter"_association_list>  --:80:
                                                        -- VHDL 7.3.3 Ada 6.4
```

```
                                --
                    ----------------
--:*80:             --
                    <association_list> ::=          -- VHDL 4.3.3.2 Ada 6.4
                         <association_element> {, <association_element> }
                                                      --:81,81:

                            -- SEM: Corresponds formal generic, port, or
                            --      parameter names with actual names or
                            --      expressions.

--:*81:
                              <association_element> ::=   -- VHDL 4.3.3.2 Ada 6.4
                                  [ <formal_part> => ] <actual_part>  --:82,84:

                                       -- SEM: Associates actual part with
                                       --      interface element in subprogram,
                                       --      component, entity, or block decl.
                                       --      See pg. 4-12 for details.

--:*82:
                                    <formal_part> ::=     -- VHDL 4.3.3.2 Ada 6.1
                                        <formal_designator>
                                        | <"function"_name> ( <formal_designator>
                                                               --:83,34,83:
--:*83:
                                      <formal_designator> ::= <"generic"_name>
                                                              | <"port"_name>
                                                              | <"parameter"_name>
                                          --:34,34,34:^  -- VHDL 4.3.3.2

--:*84:
                                    <actual_part> ::=     -- VHDL 4.3.3.2
                                        <actual_designator>
                                        | <"function"_name> ( <actual_designator> )
                                                                 --:85,34,85:

--:*85:
                                       <actual_designator> ::= <expression>
                                                              | <"signal"_name>
                                                              | <"variable"_name>
                                                              | open
                                         --:27,34:,34:^  -- VHDL 4.3.3.2

  ----------------

--:*86:
              <concurrent_signal_assignment_statement> ::=   -- VHDL 9.5
                    [ <label> :] <conditional_signal_assignment>   --:75,87:
                  | [ <label> :] <selected_signal_assignment>      --:75,104:
```

```
                    -- SEM: represents an equivalent process statement that
                    --      assigns values to signals.  See pgs 9-6 and 9-7.

--:*87:
                <conditional_signal_assignment> ::=      -- VHDL 9.5.1
                    <target> <= <options> <conditional_waveforms>
                                                         --:88,100,101:

                        -- SEM: represents a process statement in which the
                        --      signal transform is an "if statement".
                        --      See example on page 9-9.

--:*88:
                    <target> ::= <name> | <aggregate>    -- VHDL 8.3
                                                         --:34,89:

--:*89:
                        <aggregate> ::=             -- VHDL 7.3.2 Ada 4.3
                            ( <element_association>         --:90:
                              {, <element_association> })   --:90:

--:*90:
                              <element_association> ::=
                                 [ <choices> => ] <expression>   --:91,27:
                                                 -- VHDL 7.3.2 Ada 4.3

--:*91:
                                   <choices> ::= <choice> { | <choice> }
                                                 -- VHDL 7.3.2 Ada 4.3
                                                 --:92,92:

--:*92:
                                     <choice> ::= <simple_expression>
                                                 | <discrete_range>
                                                 | <"element"_simple_name>
                                                 | others
                                                        --:29,93,178:
                                                 -- VHDL 7.3.2 Ada 3.7.3

--:*93:
                                        <discrete_range> ::=  -- VHDL 3.2.1
                                           <"discrete"_subtype_indication>
                                           | <range>
                                             --                --:23,94:
          ------------------------------------
--:*94:   --
          <range> ::= <"range"_attribute_name>      -- VHDL 3.1 Ada 3.5
                    | <simple_expression> <direction> <simple_expression>
                                                 --:29,99,29:

--:*95:
```

```
                    <attribute_name> ::=          -- VHDL 6.6 Ada 4.1.4
                       <prefix> ' <attribute_designator>
                                    [( "static"_expression> )]
                                                  --:96,98,27:
--:*96:
                       <prefix> ::= <name> | <function_call>    --:34,97:
                                                  -- VHDL 6.1 Ada 4.1

--:*97:
                         <function_call> ::=      -- VHDL 7.3.3 Ada 6.4
                          <"function"_name> [( <actual_parameter_part> )]
                                                  --:34,79:

--:*98:
                        <attribute_designator> ::= <"attribute"_simple_name>
                                       --:178:    -- VHDL 6.6 Ada 4.1.4

--:*99:
                      <direction> ::= to | downto   -- VHDL 3.1

--:*100:
             <options> ::= [ guarded ][ transport ]   -- VHDL 9.5

                  -- SEM: "Guarded" specifies that the signal assignment
                  --      statement executes when a signal guard changes from
                  --      FALSE to TRUE, or when an event occurs on one of its
                  --      while the signal is TRUE.
                  --      "Transport"'specifies that the signal assignment has
                  --      transport delay.

--:*101:
             <conditional_waveforms> ::= { <waveform> when <condition> else }
                                         <waveform>
                                               -- VHDL 9.5.1  --:102,76,102:

--:*102:
                <waveform> ::= <waveform_element> {, <waveform_element>
                                                  -- VHDL 8.3  --:103,103:

--:*103:
                <waveform_element> ::=            -- VHDL 8.3.1
                    <"value"_expression> [ after <"time"_expression> ]
                    | null [ after <"time"_expression> ]     --:27,27,27:

            --   <condition> ::= -----------------^    --:76:


--:*104:
              <selected_signal_assignment> ::=       -- VHDL 9.5.2
                  with <expression> select                        --:27:
                     <target> <= <options> <selected_waveforms> ;
```

```
                                                    --:88,100,105:

--:*105:
                  <selected_waveforms> ::=          -- VHDL 9.5.2
                      { <waveform> when <choices> , }    --:102,91:
                        <waveform> when <choices>        --:102,91:

  ----------------

--:*106:
              <component_instantiation_statement> ::=   -- VHDL 9.6
                  <"instantiation"_label> :             --:75:
                      <"component"_name>                --:34:
                          [ <generic_map_aspect> ]      --:107:
                          [ <port_map_aspect> ]         --:108:

                      -- SEM: defines a subcomponent of the design entity in
                      --      which it appears and associates signals with the
                      --      ports of that subcomponent.  This subcomponent
                      --      is one instance of a class of components defined
                      --      by a corresponding component declaration.
                      --      "Component name" must be same as name declared
                      --      in component declaration.

--:*107:
                  <generic_map_aspect> ::=          -- VHDL 5.2.1.2
                      generic map ( <"generic"_association_list> )  --:80:

                          -- SEM: can specify value of generic constant in
                          --      <"generic"_interface_list>.
                          --      Generic map aspect associates a single actual
                          --      with each local generic in corresponding
                          --      component declaration.

--:*108:
                  <port_map_aspect> ::=             -- VHDL 5.2.1.2
                      port map ( <"port"_association_list> )       --:80:

                          -- SEM: associates a single actual with each local
                          --      port in corresponding component declaration.

                          -- Note: A configuration specification can bind a
                          --       component instance to a design entity and
                          --       associate local generics and ports of the
                          --       component with formal generics and ports
                          --       of that entity.

  ----------------

--:*109:
```

```
<generate_statement> ::=                    -- VHDL 9.7
     <"generate"_label> :                                  --:75:
         <generation_scheme> generate                      --:110:
             [ <concurrent_statement> ]                    --:70:
         end generate [ <"generate"_label> ]               --:75:

            -- SIM: provides iterative or conditional elaboration
            --      of a portion of the description.

            -- Note: <concurrent_statement> is a recursive
            --       nonterminal.   ---^  --:70:

--:*110:
          <generation_scheme> ::=               -- VHDL 9.7
              for <"generate"_parameter_specification>   --:111:
              | if <condition>                           --:76:

--:*111:
                  <parameter_specification> ::=  -- VHDL 8.8
                       <identifier> in <discrete_range>  --:49,93:

                       -- Note: A specification associates additional
                       --       information with a previously declared
                       --       entity.

  ----------------

--:*112:
          <block_statement> ::=                     -- VHDL 9.1 Ada 5.6
              <"block"_label> :                                 --a --:75:
                  block [ <"guard"_expression>                      --:27:
                     <block_header>                                 --:113:
                     <block_declarative_part>                   --a --:114:
                  begin                                         --a
                     <block_statement_part>                     --a --:115:
                  end block [ <"block"_label> ];                --a --:75:

                    -- SEM: defines an internal block representing a portion
                    --      of the design and may be hierarchically nested.

--:*113:
                  <block_header> ::=                -- VHDL 9.1
                      [ <generic_clause>                        --:13:
                      [ <generic_map_aspect> ;]]                --:107:

--:*114:
                    <block_declarative_part> ::=  -- VHDL 9.1
                        { <block_declarative_item> }      --:68:

                            -- Note: <block_declarative_item> is
                            --       recursive.
```

```
--:*115:
                    <block_statement_part> ::=     -- VHDL 9.1
                        { <concurrent_statement> }          --:70:

                            -- Note: this is another recursive part of
                            --       nonterminal <concurrent_statement>.

  ------------------------------------------------------------------------

--:*116:
              <process_statement> ::=                  -- VHDL 9.2
                   [ <"process"_label> :]                          --:75:
                         process [( <sensitivity_list> )]         --:117:
                            <process_declarative_part>            --:118:
                         begin
                            <process_statement_part>              --:120:
                         end process [ <"process"_label> ];       --:75:

                                    -- SEM: defines an independent sequential
                                    --      process representing behavior of
                                    --      a portion of the design.

                                    -- SEM: A passive process contains no
                                    --      signal assignment statement and
                                    --      may appear in an entity statement.
                                    --      part of an entity declaration.

--:*117:
                                  <sensitivity_list> ::=    -- VHDL 8.1
                                      <"signal"_name> {, <"signal"_name> }
                                                            --:34,34:

                                     -- SEM: If <sensitivity_list> follows
                                     --      "process", then process statement
                                     --      contains an implicit wait
                                     --      statement as its last statement
                                     --      of the form:
                                     --        wait on <sensitivity_list>;
                                     --      Only static signal names may
                                     --      appear in sensitivity list.

--:*118:
                               <process_declarative_part> ::=    -- VHDL 9.2
                                   { <process_declarative_item> }   --:119:

--:*119:
                               <process_declarative_item> ::=    -- VHDL 9.2
                                   <subprogram_declaration>      --:122:
                                   | <subprogram_body>           --:127:
                                   | <type_declaration>          --:130:
```

```
                                | <subtype_declaration>       --:166:
                                | <constant_declaration>      --:167:
                                | <variable_declaration>      --:168:
                                | <file_declaration>          --:169:
                                | <alias_declaration>         --:172:
                                | <attribute_declaration>     --:173:
                                | <attribute_specification>   --:174:
                                | <use_clause>                --:180:

--:*120:
                          <process_statement_part> ::=      -- VHDL 9.2
                              { <sequential_statement> }    --:121:

                                  --         *** Important point ***
                                  -- SEM: This group of sequential statements
                                  --      executes repetitively.  Immediately
                                  --      after the last statement executes,
                                  --      the first statement starts
                                  --      executing again.

--:*121:
                                <sequential_statement> ::=  -- VHDL 8 Ada 5.1
                                    <wait_statement>              --:196:
                                    | <assertion_statement>       --:75a:
                                    | <signal_assignment_statement>  --:200:
                                    | <variable_assignment_statement> --:201:
                                    | <procedure_call_statement>      --:78:
                                    | <if_statement>      --:202: --a,a^,a^
                                    | <case_statement>    --:204     --a
                                    | <loop_statement>    --:206:    --a
                                    | <next_statement>    --:208:
                                    | <exit_statement>    --:209:    --a
                                    | <return_statement>  --:210:    --a
                                    | <null_statement>    --:211:    --a

  --       *** Important Note on Implicit Wait Statement of a Process. ***

  -- Note: A process that has an explicit sensitivity list always has exactly
  --       one (implicit) wait statement in it, and that wait statement appears
  --       at the end of the sequence of statements in the process statement
  --       part.  Thus, a process with a sensitivity list always waits at the
  --       end of its statement part; any event on a signal named in the
  --       sensitivity list will cause such a process to execute from the
  --       beginning of its statement part down to the end, where it will wait
  --       again.  Such a process executes once through at the beginning of
  --       simulation, suspending for the first time when it executes the
  --       implicit wait statement.

  -- SIM:  Every signal assignment statement in a process statement defines a
  --       set of "drivers" for certain scaler signals.  There is a single
  --       driver for a given scalar signal S in a process statement provided
```

```
  --       that there is at least one signal assignment statement in that
  --       process statement, and the longest static prefix of the target
  --       signal of that signal assignment statement denotes S, or denotes
  --       a composite signal of which S is a subelement.  Each such signal
  --       assignment statement is said to be associated with that driver.
  --       Execution of a signal assignment statement affects only the
  --       associated driver(s).
  --       (The above is quoted from page 9-4 of the VHDL Reference Manual.
  --       Overlooking these details can cause lots of waisted time trying
  --       to get process statements to work right.)

  ------------------------------------------------------------------------
  -- Note: The following defines the subprogram structure.
  --       (All subprograms can be called recursively.)

--:*122:
            <subprogram_declaration> ::= <subprogram_specification> ;
                                            --:123:  -- VHDL 2.1 Ada 6.1

                 -- SEM: defines subprogram's calling conventions.
                         A subprogram can be a procedure or a function.

--:*123:
            <subprogram_specification> ::=          -- VHDL 2.1 Ada 6.1
                procedure <designator> [( <formal_parameter_list> )]
              | function <designator> [( <formal_parameter_list> )]
                    return <type_mark>
                               --:124,125,124,125,24:

--:*124:
                          <designator> ::= <identifier> | <operator_symbol>
                                        --:49,126:   -- VHDL 2.1 Ada 6.1

                             -- SEM: A "procedure" designator is "always"
                             --      an <identifier>.
                             --      A "function" designator is either an
                             --      <identifier> or an <operator symbol>.

                             -- Note: Next revision of VHDL should consider
                             --       distinguishing <"procedure"_designator>
                             --       from <"function"_designator> to capture
                             --       more semantic information in the B.N.F.
                             --       representation of the VHDL language.

--:*125:
                     <formal_parameter_list> ::= "parameter"_interface_list
                                              --:17:    -- VHDL 2.1.1 Ada 6.4

                            -- SEM: may be constants, variables, or signals.
                            --      Mode of parameter determines how it may be
                            --      accessed.  Only in, inout, and out modes
```

```
                        --      are allowed.
                        --      (Mode and class may determine how access
                        --      is implemented. See p. 2-2 for details.)
                        --    If mode in and no object class, then signal.
                        --    If mode inout or out and no object class,
                        --    then parameter is a variable.

                    -- Note: <interface_list> ::=  ------^ --:17:

                    -- Note: <type_mark> ::=  -----------^ --:24:
                    --       (This is the subtype of the returned value)

--:*126:
                    <operator_symbol> ::= <string_literal>   --:39:
                                                        -- VHDL 2.1 Ada 6.1

  ------------------------------------------------------------------------

--:*127:
              <subprogram_body> ::=                      -- VHDL 2.2 Ada 6.3
                  <subprogram_specification> is            --:123:  --a
                      <subprogram_declarative_part>       --:128:  --a
                  begin                                             --a
                      <subprogram_statement_part>          --:195:  --a
                  end [ <designator> ];                    --:124:  --a
                          -- SEM: defines subprogram's execution.

                    -- Note: <subprogram_specification> :: -----^ --:123:

--:*128:
                       <subprogram_declarative_part> ::=        -- VHDL 2.2
                          { <subprogram_declarative_item> }     --:129:

--:*129:
                            <subprogram_declarative_item> ::=   -- VHDL 2.2
                                <subprogram_declaration>        --:122:
                                | <subprogram_body>             --:127:
                                | <type_declaration>            --:130:
                                | <subtype_declaration>         --:166:
                                | <constant_declaration>        --:167:
                                | <variable_declaration>        --:168:
                                | <file_declaration>            --:169:
                                | <alias_declaration>           --:172:
                                | <attribute_declaration>       --:173:
                                | <attribute_specification>     --:174:
                                | <use_clause>                  --:180:

                                -- Note: <subprogram_body> is recursive.

      -- Note: See pages 2-5 and 2-6 on subprogram and operator overloading.
```

```
---------------- ---------------- ---------------- ---------------

   -- Note: The following make up the above <subprogram_declarative_item>
   --       and <process_declarative_item>.
   -- Note: <entity_declarative_item> and <block_declarative_item> consist
            of the following (plus some others listed after the following).

--:*130:
          <type_declaration> ::=                        -- VHDL 4.1 Ada 3.3.1
              <full_type_declaration> | <incomplete_type_declaration>
                                                        --:131,165:

--:*131:
              <full_type_declaration> ::=               -- VHDL 4.1 Ada 3.3.1
                   type <identifier> is <type_definition>    --:49,132:

--:*132:
                   <type_definition> ::=                -- VHDL 4.1 Ada 3.3.1
                       <scalar_type_definition>             --:133:
                       | <composite_type_definition>        --:154:
                       | <access_type_definition>           --:163:
                       | <file_type_definition>             --:164:

--:*133:
                   <scalar_type_definition> ::=         -- VHDL 3.1
                     <enumeration_type_definition> | <integer_type_definition>
                     <floating_type_definition>   | <physical_type_definition>
                                                        --:134,139,141,142:

--:*134:
                     <enumeration_type_definition> ::=  -- VHDL 3.3.1 Ada 3.5.1
                         ( <enumeration_literal> {, <enumeration_literal> }]
                                                                -:135,135:
                              -- SEM: Predefined Enumeration Types are
                              --      CHARACTER, BIT, BOOLEAN, and
                              --      SEVERITY_LEVEL in package STANDARD.

--:*135:
                                <enumeration_literal> ::=    -- VHDL 3.3.1 Ada 3.5.1
                                    <identifier> | <character_literal>  --:49,136:

--:*136:
                                   <character_literal> ::= ' <graphic_character> '
                                                 --:137:    -- VHDL 13.5 Ada 2.5

--:*137:
                                   <graphic_character> ::=   -- VHDL 13.1 Ada 2.1
                                      <basic_graphic_character>    --:138:
                                      <lower_case_letter>          -- See note.
                                      <other_special_character>    -- See note.
```

```
                                -- Note: <lower_case_letter> and
                                -- <other_special_character> are
                                -- lexical representations of
                                -- terminals.

--:*138:
                        <basic_graphic_character> ::=
                            <upper_case_letter> | <digit>
                            | <special_character> | <space_character>
                                                -- VHDL 13.1 Ada 2.1

                            -- Note: <upper_case_letter>, <digit>,
                            -- and <special_character> are lexical
                            -- representations of terminals.

                            -- Note: Further breakdown of
                            -- <graphic_character> is done by lexical
                            -- processing (not in this syntax summary)

--:*139:
                    <integer_type_definition> ::= <range_constraint>
                                        --:140:    -- VHDL 3.1.2 Ada 3.5.4

                        -- SEM: INTEGER is the only predefined integer
                        --      type.
                        --      Integer literals are of type
                        --      universal_integer.

--:*140:
                        <range_constraint> ::= range <range>   --:94:
                                                -- VHDL 3.2.2 Ada 3.7

--:*141:
                    <floating_type_definition> ::= <range_constraint>  ----^
                                                -- VHDL 3.1.4
                        -- SEM: Type REAL is the only predefined floating
                        --      point type.  Guaranteed range is -1E38 to
                        --      +1E38.

--:*142:
                    <physical_type_definition> ::=   -- VHDL 3.1.3
                         <range_constraint>                         --:140:
                             units
                                <base_unit_declaration>             --:143:
                                { <secondary_unit_declaration> }    --:144:
                             end units
                                        -- Note: No ";" after end units.
                                        --   This is different from normal.

                                -- SEM: TIME is the only predefined
                                --      physical type with a default base
```

```
                                    --      unit of 1 femtosecond (1E-15).
                                    --      Guaranteed range is -2147483647
                                    --      to +2147483467 ascending.

--:*143:
                                <base_unit_declaration> ::= <identifier>
                                                    -- VHDL 3.1.3  --:49:

--:*144:
                                <secondary_unit_declaration> ::=
                                    <identifier> = <physical_literal>;
                                                    -- VHDL 3.1.3 --:49,145:

--:*145:
                                    <physical_literal> ::=  -- VHDL 3.1.3
                                        [ <abstract_literal> ] <"unit"_name>
                                                                --:146,34:
--:*146:
                                        <abstract_literal> ::=  -- VHDL 13.4
                                           <decimal_literal> | <based_literal>
                                                    --    --:147,150:  --
            ------------------------------------------------------------
                ------------------------------------
--:*147:         --
                <decimal_literal> ::= <integer> [ , <integer> ][ <exponent> ]
                                    --:148,148,149:  -- VHDL 13.4.1 Ada 2.4.1

--:*148:
                    <integer> ::= <digit> { [ <underline> ] <digit> }
                                                    -- VHDL 13.4.1 Ada 2.4.1

                            -- Note: <digit> and <underline> are lexical
                            -- terminals.

--:*149:
                    <exponent> ::= E [ + ] <integer> | E - <integer>
                                    --:148,148:      -- VHDL 13.4.1
                                        -- Not: <integer> is recursive.
--:*150:    --
            <based_literal> ::=                     -- VHDL 13.4.2 Ada 2.4.2
               <base> # <based_literal> [ . <based_integer> ] # [ <exponent> ]
                                            --:151,150,150,149:

                                    -- Note: <based_literal> is recursive.

--:*151:
               <base> ::= <integer>    --:148:        -- VHDL 13.4.2 Ada 2.4.2

--:*152:
                 <based_integer> ::=                -- VHDL 13.4.2 Ada 2.4.2
                      <extended_digit> { [ <underline> ] <extended_digit> ]
```

```
                                                --:153,_,153:

--:*153:
                    <extended_digit> ::= <digit> | <letter>
                                                -- VHDL 13.4.2 Ada 2.4.2

                                        -- Note: <digit> and <letter> are
                                        -- lexical representations of terminals.

  -----------------

--:*154:
                  <composite_type_definition> ::=    -- VHDL 3.2
                      <array_type_definition> | <record_type_definition>
                                          --:155,160:

--:*155:
                      <array_type_definition> ::=    -- VHDL 3.2.1 Ada 3.6
                          <unconstrained_array_definition>    --:156:
                          | <constrained_array_definition>    --:158:

                               -- SEM: Predefined array types are STRING and
                               --      BIT_VECTOR in package standard.

--:*156:
                          <unconstrained_array_definition> ::=
                              array ( <index_subtype_definition>
                                 { , <index_subtype_definition> })
                                 of <"element"_subtype_indication>  ----^
                                   --:157,157,23:    -- VHDL 3.2.1 Ada 3.6

--:*157:
                              <index_subtype_definition> ::=
                                  <type_mark> range <>          --:24:
                                                -- VHDL 3.2.1 Ada 3.6
                                      -- Note: Make sure you differentiate
                                      -- between <> terminal and <text>
                                      -- nonterminal in your parser.

--:*158:
                          <constrained_array_definition> ::=
                              array <index_constraint> of        --:159:
                                 <"element"_subtype_indication>   --:23:
                                                -- VHDL 3.2.1 Ada 3.6

--:*159:
                                 <index_constraint> ::=
                                  ( <discrete_range> { , <discrete_range> })
                                                --:93,93:
                                                -- VHDL 3.2.1 Ada 3.6
                                       -- SEM: determines array bounds.
```

```
--:*160:
                    <record_type_definition> ::=   -- VHDL 3.2.2 Ada 3.7
                        record
                           <element_declaration>        --:161:
                           { <element_declaration> }    --:161:
                        end record
                                  -- Note: No ";" after "end record".

--:*161:
                       <element_declaration> ::=   -- VHDL 3.2.2
                         <identifier_list> : <element_subtype_definition> ;
                                                          --:162:

--:*162:
                         <element_subtype_definition> ::=  -- VHDL 3.2.2
                                       <subtype_indication>     --:23:

  -----------------

--:*163:
                 <access_type_definition> ::= access <subtype_indication>
                                                  --:23:
                                                 -- VHDL 3.3 Ada 3.8
                      -- SEM: Access to objects created by evaluation of
                      --      allocators is achieved by an access value
                      --      returned by an allocator.  The access value
                      --      designates the object.  (See 3.3.2 and 7.3.6
                      --      on allocators.)
                      --      For each access type, a deallocation operation
                      --      is "implicitly" declared immediately following
                      --      the full type declaration for the type.  (Thus,
                      --      the storage occupied by a designated object can
                      --      be explicitly deallocated.)

--:*164:
                 <file_type_definition> ::= file of <type_mark>   --:24:
                                                 -- VHDL 3.4
                      -- SEM: define objects representing files in the host
                      --      environment.  The value of a file object is the
                      --      sequence of values contained in the host system
                      --      environment.
                      --      <type_mark> defines the subtype of values in the
                      --      file.  Base type of this subtype must not be a
                      --      file type or an access type.  If the base type
                      --      is a composite type, it must not contain a
                      --      subelement of an access type.  If the base type
                      --      is an array type, it must be a one-dimensional
                      --      array type.
                      --      File operations for objects of file type include
                      --        procedure READ (F: in FT; VALUE: out TM);
```

```
                    --        procedure WRITE (F: out FT; VALUE: in TM);
                    --        function ENDFILE (F: in FT) return BOOLEAN;

  -----------------

--:*165:
            <incomplete_type_definition> ::= type <identifier> ;    --:49:
                                                        -- VHDL 3.3.1 Ada 3.8.1
                 -- SEM: For each incomplete type declaration there must be
                 --      a corresponding full type declaration with the
                 --      same identifier (which occurs later and immediately
                 --      within the same declarative part).

  ----------------- ---------------- ------------------- -----------------

--:*166:
          <subtype_declaration> ::=                     -- VHDL 4.2 Ada 3.3.2
              subtype <identifier> is <subtype_indication>     --:49,23:

--:*167:
          <constant_declaration> ::=                    -- VHDL 4.3.1.1 Ada 3.2
              constant <identifier_list> : <subtype_indication>       --:26,23:
                                         [ := <expression> ] ;        --:27:
                                                          ---------^
                 -- SEM: If no default value for constant declaration in
                 --      package declaration is present, then corresponding
                 --      full constant declaration must appear in body of
                 --      package.

--:*168:
          <variable_declaration> ::=                    -- VHDL 4.3.1.3
              variable <identifier_list> : <subtype_indication>       --:26,23:
                                         [ := <expression> ] ;
                                                          ---------^
                  -- SEM: If initial value exists, then the initial
                  --      value of the expression is determined each time the
                  --      variable declaration is elaborated.
                  --      Default initial value for a variable of scalar type
                  --      has value T'Left.
                  --      Default initial value of a variable of an access type
                  --      has value null for that type.

--:*169:
          <file_declaration> ::=                        -- VHDL 4.3.2
              file <identifier> : <subtype_indication> is [ mode ]    --:49,23:
                                      <file_logical_name> ;       --:171:

--:*170:
              <mode> ::= in | out | inout | buffer | linkage
                                                        -- VHDL 4.3.3 Ada 6.1
                   -- SEM: If no mode explicitly given in interface
```

```
                --      declaration, then mode is "in".

--:*171:
        <file_logical_name> ::= <"string"_expression>   -- VHDL 4.3.2
                                                        --:27:

        -- Note: <subtype_indication> ::= -----------^  --:23:

--:*172:
      <alias_declaration> ::=                       -- VHDL 4.3.4
          alias <identifier> : <subtype_indication> is <name> ;  ----^
                                                        --:49,23,34:

             -- SEM: declares an alternate name for an existing object.
             --      <name> must be a static name (6.1).

--:*173:
      <attribute_declaration> ::=                   -- VHDL 4.4
          attribute <identifier> : <type_mark> ;          --:49,24:

             -- SEM: An attribute is a value, function, type, range,
             --      signal, or constant that may be associated with one
             --      or more entities in a description.  Attributes can
             --      be predefined or user-defined. (See Chapt 14.)
             --      Predefined attribute signals cannot be updated.
             --      User-defined attributes are constants of arbitrary
             --      type.
             --      An attribute may be associated with an entity,
             --      architecture, configuration, procedure, function,
             --      package, type, subtype, constant, signal, variable,
             --      component, or label.
             --      <type_mark> cannot be an access or file type.

--:*174:
      <attribute_specification> ::=                 -- VHDL 5.1
          attribute <attribute_designator> of <entity_specification> is
                                              <expression> ;
                                                --:98,175,27:

             -- SEM: associates a user-defined attribute with one or more
             --      entities and defines the value of that attribute for
             --      those entities.

             -- SEM: ERROR if attribute associates more than once with
             --      a given entity.

                          ------------------------------------^

--:*175:
        <entity_specification> ::=                  -- VHDL 5.1
            <entity_name_list> : <entity_class>     --:176,179:
```

```
--:*176:
                <entity_name_list> ::=                  -- VHDL 5.1
                    <entity_designator> {, <entity_designator> }
                    | others
                    | all
                                --:177,177:^

--:*177:
                  <entity_designator> ::= <simple_name> | <operator_symbol>
                                          --:178,126:

                       -- SEM: An entity designator that is an operator
                       --      symbol is used to associate an attribute
                       --      with an overloaded operator.

                                                  -- VHDL 5.1

--:*178:
                       <simple_name> ::= <identifier>  -- VHDL 6.2 Ada 4.1
                                                                  --:49:

--:*179:
                  <entity_class> ::=                    -- VHDL 5.1
                      entity        | architecture    | configuration
                      | procedure   | function        | package
                      | type        | subtype         | constant
                      | signal      | variable        | component
                      | label

--:*180:
          <use_clause> ::=                              -- VHDL 10.4 Ada 8.4
              use <selected_name> { , <selected_name> } ;     --:181,181:

--:*181:
                <selected_name> ::= <prefix> . <suffix>     --:96,182:
                                    --------------------^
                                                  -- VHDL 6.3 Ada 4.1.3

--:*182:
                   <suffix> ::= <simple_name>  --:178:-- VHDL 6.3 Ada 4.1.3
                                | <character_literal>     --:136:
                                | <operator_symbol>       --:126:
                                | all

          -- Note: (Number 183 has beed deleted on purpose.)
          --       <character_literal> ::= ------------^     --:136:
                                                  -- VHDL 13.5 Ada 2.5

  ------------------------------------------------------------------
   -- Note: The following make up the rest of <entity_declarative_item>.
```

```
--:*184:
          <signal_declaration> ::=                    -- VHDL 4.3.1.2
              signal <identifier_list> : <subtype_indication>      --:26,23:
                                     [ <signal_kind> ] [ := <expression> ] ;
                                                                  --:185,27:

                  -- SEM: If resolution function appears in subtype indication
                  --      of signal declaration, then signal is called a
                  --      resolved signal.

--:*185:
              <signal_kind> ::= register | bus        -- VHDL 4.3.1.2

                  -- SEM: means declared signals are guarded by signal kind.

  -------------

--:*186:
          <disconnection_specification> ::=           -- VHDL 5.3
              disconnect <guarded_signal_specification> after
                                          <"time"_expression> ;
                                                          --:187,27:^

                 -- SEM: defines the time delay used in the implicit
                 --      disconnection of drivers of a guarded signal within
                 --      a guarded signal assignment.

                 --      Time expression must be static an non-negative.

--:*187:
              <guarded_signal_specification> ::=      -- VHDL 5.3
                  <"guarded"_signal_list> : <type_mark>         --:188,24:
                                             -------------^

--:*188:
                  <signal_list> ::=                   -- VHDL 5.3
                      <"signal"_name> { , <"signal"_name> }     --:34,34:
                      | others
                      | all

                         -- SEM: identifies signals for which implicit
                         --      disconnection delay is to be defined.
                         --      (See details on page 5-7.)

                         -- SEM: ERROR if more than one disconnection
                         --      specification applies to drivers of the
                         --      same signal.

                           -- Note: Define <name> later.
```

```
-----------------------------------------------------------------
  -- Note: The following make up the rest of <block_declarative_item>.

--:*189:
          <component_declaration> ::=                    -- VHDL 4.5
               component <identifier>                        --:49:
                  [ <"local"_generic_clause> ]   -----^     --:13:
                  [ <"local"_port_clause> ]      -----^     --:14:
               end component ;

                  -- SEM: defines a virtual design entity interface that may be
                  --      used in a component instantiation statement.
                  --      A component configuration or configuration
                  --      specification can associate a component instance with
                  --      a design entity that resides in a library.

  --------------

--:*190:
          <configuration_specification> ::=              -- VHDL 5.2
               for <component_specification> use <binding_indication> ;
                                                              --:191,193:

                     -- SEM: associates binding information with component
                     --      labels representing instances of a given
                     --      component.

--:*191:
                  <component_specification> ::=          -- VHDL 5.3
                      <instantiation_list> : <"component"_name>  --:192,34:

                        -- SEM: identifies component instances.

--:*192:
                      <instantiation_list> ::=           -- VHDL 5.2
                        <"instantiation"_label> { , <"instantiation"_label> }
                        | others                              --:75,75:^
                        | all

                             -- SEM: identifies those entities with which
                             --      binding information is to be associated,
                             --      (as defined at bottom of page 5-3).

--:*193:
                <binding_indication> ::=                 -- VHDL 5.2.1
                    <entity_aspect>                      --:194:
                    [ <generic_map_aspect> ]   -----^    --:107:
                    [ <port_map_aspect> ]      -----^    --:108:

                         -- SEM: associates component instances with a
                         --      particular design entity.
```

```
--:*194:
                    <entity_aspect> ::=                    -- VHDL 5.2.1.1
                       entity <"entity"_name> [( <"architecture"_identifier> )]
                       | configuration <"configuration"_name>    --:34,49,34:
                       | open

                            -- SEM: identifies design entity to be associated
                            --      with component instances.  It may also
                            --      specify which binding is to be deferred.


   -- Note: End of declarations.
  ------------------------------------------------------------------------
   -- Note: The following make up the <subprogram_statement_part> of
   --       <subprogram_body>.

--:*195:
          <subprogram_statement_part> ::=              -- VHDL 2.2
              { <sequential_statement> }                     --:121:

  ------------------------------------------------------------------------
   -- Note: The following make up <sequential_statement> which can be found
   --       embedded in the <process_statement> and <subprogram_body>.

--:*196:
           <wait_statement> ::=                        -- VHDL 8.1
              wait [ <sensitivity_clause> ] [ <condition_clause> ]
                   [ <timeout_clause> ] ;
                                        --:197,198,199:

                      -- SEM: causes suspension of a process statement or a
                      --      procedure.

--:*197:
                    <sensitivity_clause> ::= on <sensitivity_list>   --:117:
                                                 -- VHDL 8.1
                      -- SEM: defines the set of signals to which the wait
                      --      statement is sensitive.  Each signal name
                      --      must be a static signal name.

--:*198:
                    <condition_clause> ::= until <condition>         --:76:
                                                 -- VHDL 8.1
                      -- SEM: specifies condition that must be met for
                      --      for process to continue execution.

--:*199:
                    <timeout_clause> ::= for <"time"_expression>     --:27:
                                                 -- VHDL 8.1
                      -- SEM: specifies maximum time that process will remain
```

```
                        --      suspended at this wait statement.

                        -- SEM: ERROR if time expression evaluates to negative.

    ------------

        --  <assertion_statement> ::=  --------^  --:75a:

              -- SEM: checks that a specified condition is true and reports an
              --      error if it is not.

    ------------

--:*200:
          <signal_assignment_statement> ::=            -- VHDL 8.3
              <target>  <=  [ transport ] <waveform> ;     --:88,102:

              -- SEM: modifies the projected output waveforms contained in the
              --      drivers of one or more signals.  (See also 9.2.1).
              --      If <target> is a name, then name must denote a signal.
              --      If <target> is an aggregate, then type of aggregate must
              --      be determinable from the context.
              --      Transport specifies that the delay associated with the
              --      first waveform element is transport delay.
              --      If transport is absent, then delay is inertial delay (a
              --      characteristic of switching circuits).

    ------------

--:*201:
          <variable_assignment_statement> ::=          -- VHDL 8.4 Ada 5.2
              <target> <= <expression> ;     ----------^  --:88,27:  --***

              -- SEM: replaces current value of variable with new value specified
              --      by expression. (It must be the same type.)

    ------------

        --  <procedure_call_statement> ::=  ---^ --:78:  -- VHDL 8.5 Ada 6.4

    ------------

--:*202:
          <if_statement> ::=                           -- VHDL 8.6 Ada 5.3
              if <condition> then                 ------^  --:76:
                    <sequence_of_statements>             --:203:
              { elsif <condition> then                  --:76:
                    <sequence_of_statements> }           --:203:
              [ else
                    <sequence_of_statements> ]           --:203:
              end if ;
```

```
                    -- SEM: <condition> must be of type BOOLEAN.

--:*203:
                 <sequence_of_statements> ::=        -- VHDL 8   Ada 5.1
                    { <sequential_statement> }          --:121:

                        -- Note:  <sequential_statement> is recursive.

  -------------

--:*204:
          <case_statement> ::=                          -- VHDL 8.7 Ada 5.4
              case <expression> is                        --:27:
                 <case_statement_alternative>             --:205:
                 { <case_statement_alternative> }         --:205:
              end case ;

                     -- SEM: <expression> must be of a discrete type.

--:*205:
                <case_statement_alternative> ::=    -- VHDL 8.7 Ada 5.4
                    when <choices> =>          -------^   --:91:
                      <sequence_of_statements> --------^  --:203:

                        -- Note: Another place where <sequential_statement>
                        --       is recursive.

  --------------

--:*206:
          <loop_statement> ::=                          -- VHDL 8.8 Ada 5.5
              [ <"loop"_label> : ]                          --:75:
                    [ <iteration_scheme> ] loop             --:207:
                         <sequence_of_statements>   --------^  --:203:
                    end loop [ <"loop"_label> ] ;           --:75:

--:*207:
                    <iteration_scheme> ::=          -- VHDL 8.8 Ada 5.5
                        while <condition>     -------^        --:76:
                        | for <"loop"_parameter_specification>  ---^ --:111:

                            -- SEM: A "while" iteration scheme evaluates
                            --      before each execution of sequence of
                            --      statements.
                            --      A "for" iteration scheme executes
                            --      after each evaluation of the discrete
                            --      range.

  --------------
```

```
--:*208:
          <next_statement> ::=                              -- VHDL 8.9
             next [ <"loop"_label> ] [ when <condition> ] ;
                                                  ------------^  --:75,76:
                 -- SEM: used to complete execution of one of the iterations of
                 --      an enclosing loop statement.

  --------------

--:*209:
          <exit_statement> ::=                              -- VHDL 8.10 Ada 5.7
              exit [ <"loop"_label> ] [ when <condition> ] ;
                                                  ------------^  --:75,76:
                 -- SEM: used to complete execution of an enclosing loop
                 --      statement.  Only allowed within a labeled loop.

  --------------

--:*210:
          <return_statement> ::=                            -- VHDL 8.11 Ada 5.8
              return [ <expression> ] ;                     --:27:

                 -- SEM: used to complete execution of innermost enclosing
                 --      function or procedure body.

  --------------

--:*211:
          <null_statement> ::= null ;                       -- VHDL 8.12 Ada 5.1

                 -- SEM: performs no action. (like a noop.)

  ------------------------------------------------------------------------

  -- Note:  The following non-terminals from Appendix A (Syntax Summary)
  --        of the VHDL Reference Manual (IEEE Standard 1076-1987) contains
  --        the following dangling non-terminals (i.e. they have no roots
  --        within the rest of the VHDL syntax summary).

--:*212:
     <basic_character> ::=                                  -- VHDL 13.1 Ada 2.1
         <basic_graphic_character> | <format_effector>      --:138,.: --See note.

          -- Note: <format_effector> is the lexical representation of a
          -- terminal.

          -- SEM: Each graphic character corresponds to ISO seven-bit coded
                  character set (ISO 646-1983).

          -- NOTE: As in Ada 9X, the next revision of VHDL should consider
                   extending the character set to include international
```

```
                characters (corresponding to how Ada 9X decides to do it).

--:*213:
     <declaration> ::=                                  -- VHDL 4    Ada 3.1
         <type_declaration>                                          --a --:130:
         | <subtype_declaration>                                     --a --:166:
         | <object_declaration>     -----v                          --a --:216:
         | <file_declaration>                                            --:169:
         | <interface_declaration>                                       --:19:
         | <alias_declaration>                                           --:172:
         | <attribute_declaration>                                      --:173:
         | <component_declaration>                                      --:189:
         | <entity_declaration>                                         --::11
         | <configuration_declaration>                                  --:56:
         | <subprogram_declaration>                                   --a --:122:
         | <package_declaration>                                      --a --:63:

--:*214:
     <logical_operator> ::= and | or | nand | nor | xor   -- VHDL 7.2 Ada 4.5

--:*215:
     <miscellaneous_operator> ::= ** | abs | not       -- VHDL 7.2

--:*216:
     <object_declaration> ::=                           -- VHDL 4.3.1 (Ada 3.2)
          <constant_declaration>                                    --:167:
          | <signal_declaration>                                    --:184:
          | <variable_declaration>                                  --:168:

          -- SEM: An object is an entity that contains a value of a given type.

  ------------------- (End of VHDL Syntax Summary) --------------------------
```