CV: Grammar of the VHDL Subset

Jump to:

0. Conventions

This page provides a summary of the syntax for the VHDL subset of CV. Productions are ordered in order of appearance in the VHDL Language Reference Manual (LRM). The form of a production is described by means of a context-free grammar, using Backus-Naur notation (the following is lifted from the LRM):

- Lowercased words denote syntactic categories; for example:

  formal_port_list,

- Boldface words denote language keywords for example:

  entity

- A vertical bar separates alternative items on the right hand side of a production unless it occurs right after an opening brace, in which case it stands for itself:

  letter_or_digit::=letter | digit

  choices ::= choice { **|** choice }

- Square brackets enclose optional items for example:

  return_statement ::= **return** [statement]

- Braces enclose a repeated item or items that may appear zero or more tumes:

  term ::= factor { multiplying_operator factor }

1. Design entities and configurations

1.1 Entity declarations

entity_declaration ::= **entity** identifier **is** entity_header [begin] end [entity] [entity_simple_name] **;**

1.1.1 Entity header

entity_header ::= [formal_port_clause]

port_clause ::= port (port_list) ;

1.1.1.2 Port lists

port_list ::= port_interface_list

1.2 Architecture bodies

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

1.2.1 Architecture declarative part

architecture_declarative_part ::= { block_declarative_item }

1.2.2 Architecture statement part

architecture_statement_part ::= { concurrent_statement }

---

2. Subprograms and Packages

2.5 Package declarations

package_declaration ::= **package** identifier **is** package_declarative_part **end** [package] [package_simple_name] **;**

package_declarative_part ::= { package_declarative_item }

package_declarative_item ::= type_declaration | subtype_declaration | constant_declaration | use_clause

2.6 Package bodies

package_body ::= **package body** package_simple_name **is** package_body_declarative_part **end** [package body] [package_simple_name] **;**

package_body_declarative_part ::= { package_body_declarative_item }

package_body_declarative_item ::= type_declaration | subtype_declaration | constant_declaration | use_clause

---

3. Types

3.1 Scalar types

scalar_type_definition ::= enumeration_type_definition | integer_type_definition

3.1.1 Enumeration types

enumeration_type_definition ::= **(** enumeration_literal { **,** enumeration_literal } **)** enumeration_literal ::= identifier | character_literal

3.1.2 Integer types

integer_type_definition ::= range_constraint

range_constraint ::= **range** range

range ::= simple_expression direction simple_expression

direction ::= **to** | **downto**

---

4. Declarations

declaration ::= type_declaration | subtype_declaration | object_declaration | interface_declaration | entity_declaration | package_declaration

4.1.1 Type declarations

type_declaration ::= full_type_declaration

full_type_declaration ::= **type** identifier **is** type_definition **;**

type_definition ::= scalar_type_definition

4.2 Subtype declarations

subtype_declaration ::= **subtype** identifier **is** subtype_indication **;**

subtype_indication ::= type_mark [constraint]

type_mark ::= type_name | subtype_name

constraint ::= range_constraint

4.3 Object declarations

object_declaration ::= constant_declaration | signal_declaration | variable_declaration

4.3.1.1 Constant declarations

constant_declaration ::= **constant** identifier_list **:** subtype_indication **:=** expression **;**

4.3.1.2 Signal declarations

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

4.3.1.3 Variable declarations

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

4.3.3 Interface declarations

interface_declaration ::= interface_signal_declaration interface_signal_declaration ::= [**signal**] identifier_list **:** [mode] subtype_indication [**:=** static_expression]

mode ::= **in** | **out**

4.3.3.1 Interface lists

interface_list ::= interface_element { ; interface_element }

interface_element ::= interface_declaration

---

5. Specifications

---

6. Names

name ::= simple_name | operator_symbol | selected_name

6.2 Simple names

simple_name ::= identifier

6.3 Selected names

selected_name ::= prefix **.** suffix

prefix ::= name

suffix ::= simple_name | **all**

---

7. Expressions

7.1 Relations

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

relation ::= shift_expression [relational_operator shift_expression]

shift_expression ::= simple_expression

simple_expression ::= term

term ::= factor

factor ::= primary | **not** primary

primary ::= name | literal | (expression)

7.2.2 Relational operators

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

7.3.1 Literals

literal ::= numeric_literal | enumeration_literal

numeric_literal ::= abstract_literal

---

8. Sequential statements

sequence_of_statements ::= { sequential_statement }

sequential_statement ::= wait_statement | signal_assignment_statement | variable_assignment_statement | if_statement | case_statement | null_statement

8.1 Wait statement

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

label ::= identifier

sensitivity_clause ::= **on** sensitivity_list

sensitivity_list ::= signal_name { **,** signal_name }

condition_clause ::= **until** condition

8.3 Signal assignment statement

signal_assignment_statement ::= [label **:**] target **<=** waveform **;**

target ::= name

inertial waveform ::= waveform_element

8.3.1 Waveform element

waveform_element ::= value_expression

8.4 Variable assignment statement

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

8.6 If statement

if_statement ::= [if_label **:**] **if** condition **then** sequence_of_statements { **elsif** condition **then** sequence_of_statements } [**else** sequence_of_statements] **end if** [if_label] **;**

condition ::= boolean_expression

8.7 Case statement

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

case_statement_alternative ::= **when** choices **=>** sequence_of_statements

8.8 Loop statement

loop_statement ::= [loop_label :] **while** condition **loop** sequence_of_statements **end loop** [loop_label] **;**

8.12 Null statement

null_statement ::= [label **:**] **null ;**

---

9. Concurrent statements

concurrent_statement ::= block_statement | process_statement | concurrent_signal_assignment_statement

9.1 Block statement

block_statement ::= block_label : **block** [**is**] block_header block_declarative_part **begin** block_statement_part **end block** [block_label] **;**

block_declarative_part ::= { block_declarative_item }

block_declarative_item ::= type_declaration | subtype_declaration | constant_declaration | signal_declaration | use_clause

block_statement_part ::= { concurrent_statement }

9.2 Process statement

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

process_declarative_part ::= { process_declarative_item }

process_declarative_item ::= type_declaration | subtype_declaration | constant_declaration | variable_declaration | use_clause

process_statement_part ::= { sequential_statement }

9.5 Concurrent signal assignment statement

concurrent_signal_assignment_statement ::= [label :] conditional_signal_assignment **;** | [label :] selected_signal_assignment **;**

9.5.1 Conditional signal assignment statement

conditional_signal_assignment ::= target **<=** options conditional_waveforms **;**

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

9.5.2 Selected signal assignment statement

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

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

---

10. Scope and visibility

use_clause ::= **use** selected_name { **,** selected_name } **;**

---

11. Design units and their analysis

design_file ::= design_unit { design_unit }

design_unit ::= context_clause library_unit

context_clause ::= { context_item }

context_item ::= library_clause | use_clause

library_clause ::= library logical_name_list ;

logical_name_list ::= logical_name { , logical_name }

logical_name ::= identifier

library_unit ::= primary_unit | secondary_unit

primary_unit ::= entity_declaration | package_declaration

secondary_unit ::= architecture_body | package_body

See also:

- The specification language: a comprehensive description of the specificaion language.
- Examples: a repository of small examples to get you started.

Documentation Sections: cva(1) VHDL Grammar cvc(1) Specification Language

Main Sections: Introduction Installation Documentation Examples

---

CV / Carnegie Mellon University / cmuvhdl@cs.cmu.edu / Revised December 1996