~~Section~~ Clause[1] 11

Design units and their analysis

The overall organization of descriptions, as well as their analysis and subsequent definition in a design library, are discussed in this ~~section~~ clause[2].

11.1 Design units

Certain constructs ~~may be~~ are[3] independently analyzed and inserted into a design library; these constructs are called *design units*. One or more design units in sequence comprise a *design file*.

```
design_file ::=  design_unit { design_unit }

design_unit ::=  context_clause library_unit

library_unit ::=
        primary_unit
      | secondary_unit

primary_unit ::=
        entity_declaration
      | configuration_declaration
      | package_declaration

secondary_unit ::=
        architecture_body
      | package_body
```

Design units in a design file are analyzed in the textual order of their appearance in the design file. Analysis of a design unit defines the corresponding library unit in a design library. A *library unit* is either a primary unit or a secondary unit. A secondary unit is a separately analyzed body of a primary unit resulting from a previous analysis.

The name of a primary unit is given by the first identifier after the initial reserved word of that unit. Of the secondary units, only architecture bodies are named; the name of an architecture body is given by the identifier following the reserved word **architecture**. Each primary unit in a given library must have a simple name that is unique within the given library, and each architecture body associated with a given entity declaration must have a simple name that is unique within the set of names of the architecture bodies associated with that entity declaration.

1. To conform to IEEE rules.
2. To conform to IEEE rules.
3. IR1000.4.7.

Entity declarations, architecture bodies, and configuration declarations are discussed in ~~Section~~ Clause[4] 1. Package declarations and package bodies are discussed in ~~Section~~ Clause[5] 2.

11.2 Design libraries

A *design library* is an implementation-dependent storage facility for previously analyzed design units. A given implementation is required to support any number of design libraries.

library_clause ::= **library** logical_name_list ;

logical_name_list ::= logical_name { , logical_name }

logical_name ::= identifier

A library clause defines logical names for design libraries in the host environment. A library clause appears as part of a context clause at the beginning of a design unit. There is a certain region of text called the *scope* of a library clause; this region, contained within the root declarative region (see 10.1)[6] starts immediately after the library clause, and it extends to the end of the root[7] declarative region associated with the design unit in which the library clause appears. Within this scope each logical name defined by the library clause is directly visible, except where hidden in an inner declarative region by a homograph of the logical name according to the rules of 10.3.

If two or more logical names having the same identifier (see 13.3) appear in library clauses in the same context clause, the second and subsequent occurrences of the logical name have no effect. The same is true of logical names appearing both in the context clause of a primary unit and in the context clause of a corresponding secondary unit.

Each logical name defined by the library clause denotes a design library in the host environment.

For a given library logical name, the actual name of the corresponding design ~~libraries~~ library[8] in the host environment may or may not be the same. A given implementation must provide some mechanism to associate a library logical name with a host-dependent library. Such a mechanism is not defined by the language.

There are two classes of design libraries: working libraries and resource libraries. A *working library* is the library into which the library unit resulting from the analysis of a design unit is placed. A *resource library* is a library containing library units that are referenced within the design unit being analyzed. Only one library ~~may be~~ is[9] the working library during the analysis of any given design unit; in contrast, any number of libraries (including the working library itself) may be resource libraries during such an analysis.

Every design unit except package STANDARD is assumed to contain the following implicit context items as part of its context clause:

library STD, WORK ; **use** STD.STANDARD.**all** ;

Library logical name STD denotes the design library in which package STANDARD and package TEXTIO reside; these are the only standard packages defined by the language (see ~~Section~~ Clause[10] 14). (The use clause makes all declarations within package STANDARD directly visible within the corresponding design unit; see 10.4). Library logical name WORK denotes the current working library during a given analysis.

4. To conform to IEEE rules.
5. To conform to IEEE rules.
6. LCS 3.
7. LCS 3.
8. Noted by Ashenden.
9. IR1000.4.7.
10. To conform to IEEE rules.

The library denoted by the library logical name STD contains no library units other than package STANDARD and package TEXTIO.

A secondary unit corresponding to a given primary unit ~~may only~~ must[11] be placed into the design library in which the primary unit resides.

NOTE

—The design of the language assumes that the contents of resource libraries named in all library clauses in the context clause of a design unit will remain unchanged during the analysis of that unit (with the possible exception of the updating of the library unit corresponding to the analyzed design unit within the working library, if that library is also a resource library).

11.3 Context clauses

A context clause defines the initial name environment in which a design unit is analyzed.

```
context_clause ::=  { context_item }

context_item  ::=
      library_clause
    | use_clause
```

A library clause defines library logical names that may be referenced in the design unit; library clauses are described in 11.2. A use clause makes certain declarations directly visible within the design unit; use clauses are described in 10.4.

NOTE

—The rules given for use clauses are such that the same effect is obtained whether the name of a library unit is mentioned once or more than once by the applicable use clauses, or even within a given use clause.

11.4 Order of analysis

The rules defining the order in which design units can be analyzed are direct consequences of the visibility rules. In particular:

a) A primary unit whose name is referenced within a given design unit must be analyzed prior to the analysis of the given design unit.

b) A primary unit must be analyzed prior to the analysis of any corresponding secondary unit.

In each case, the second unit *depends on* the first unit.

The order in which design units are analyzed must be consistent with the partial ordering defined by the above rules.

If any error is detected while attempting to analyze a design unit, then the attempted analysis is rejected and has no effect whatsoever on the current working library.

A given library unit is potentially affected by a change in any library unit whose name is referenced within the given library unit. A secondary unit is potentially affected by a change in its corresponding primary unit. If a library unit is changed (e.g., by reanalysis of the corresponding design unit), then all library units that are potentially affected by such a change become obsolete and must be reanalyzed before they can be used again.

11. IR1000.4.7.