False Paths Sprint-2 IEEE 1076-2019 VHDL Manual Review

Mert Ecevit

Edited at 03.12.2024

Review and Taken Notes of the Document

Project Leader: Koray Karakurt

Chapter 3 Review and Notes

3.2 Entity Declarations

- The entity declaration provides an external view of a component but does not provide information about how a component is implemented.
- An entity declaration defines the interface between a given design entity and the environment in which it is used.
- A given entity declaration may be shared by many design entities, each of which has a different architecture.
- The entity declarative part of a given entity declaration declares items that are common to all design entities whose interfaces are defined by the given entity declaration.
- The entity statement part contains concurrent statements that are common to each design entity with this interface.
- All entity statements shall be passive. Such statements may be used to monitor the operating conditions or characteristics of a design entity.

3.3 Architecture Declarations

- An architecture provides an "internal" view of an entity. An entity may have more than one architecture.
- It defines the relationships between the inputs and the outputs of a design entity which may be expressed in terms of:
  - Behavioural style
  - Data flow style
  - Structural style
- An architecture determines the function of an entity.
- It consists of a declaration section where signals, types, constants, components, and subprograms are declared, followed by a collection of concurrent statements.
- More than one architecture body may exist corresponding to a given entity declaration.
- Each declares a different body with the same interface; thus, each together with the entity declaration represents a different design entity with the same interface.
- The architecture statement part contains statements that describe the internal organization and/or operation of the block defined by the design entity.
- All of the statements in the architecture statement part are concurrent statements, which execute asynchronously with respect to one another.

3.4 Configuration Declarations

- During the design process, a designer may want to experiment with different variations of a design by selecting different architectures.
- Configurations can be used to provide fast substitutions of component instances of a structural design.

3.4.1 Block Configurations

- Blocks are used to organise a set of concurrent statements hierarchically.

3.4.2 Component Configurations

- A component declaration is required to make a design entity useable within the current design.

Chapter 4 Review and Notes

4.2 Subprogram Declarations

- Subprograms consist of procedures and functions that can be invoked repeatedly from different locations in a VHDL description
- VHDL provides two kinds of subprograms : Procedures and Functions
- Note: Different from 1076-2002 Document The properties of implicitly declared subtypes denoted by return identifier added in this section.

4.3 Subprogram Bodies

- A subprogram body specifies the execution of a subprogram
- The declaration of subprogram is optional, in absence of that subprogram specification acts as declaration
- It is an error if subprogram declarative part declares a shared variable.

4.4 Subprogram Instantiation Declarations

- Note : This section is is absent in 1076-2002 Document
- The uninstantiated subprogram name shall denote an uninstantiated subprogram declared in a subprogram declaration
- Subprogram instantiations is creating anonymous subprograms which allows you to declare and use subprograms directly within the body of another construct, such as an architecture, without needing a separate named declaration.

4.5 Subprogram Overloading

- If the two subprogram names match, the parameter set/return values have to differ. This is called overloading and is allowed for all subprograms. It is especially useful when applied to operators, which can be seen as functions with a special name
- This allows, for example, to use the conventional '+' symbol for the addition of integer values and, likewise, with bit vectors that should be interpreted as numbers

4.5.3 Signatures

- A signature distinguishes between overloaded subprograms and enumerationliterals based on their parameter and result type profiles. A signature may be used in an attribute name, entity designator or alias declaration.

4.6 Resolution Functions

- A resolution function defines how values from multiple sources, multiple drivers, are resolved into a single value.
- A type or a signal may be defined to have a resolution. This type or signal uses the resolution functions when there are multiple drivers.

4.7 Package Declarations

- The primary purpose of a package is to collect elements that can be shared among two or more design units.
- It contains some common data types, constants, and subprogram specifications.
- A package declaration declares all the names of items that will be seen by the design units that use the package

4.8 Package Bodies

- A package body contains the implementation details of the subprograms declared in the package declaration.
- A package body is not required if no subprograms are declared in a package declaration.
- The separation between package declaration and package body serves thesame purpose as the separation between the entity declaration and architecture body.

4.9 Package Instantiation Declarations

- A package with a generic list is called an uninstantiated package
- The uninstantiated package serves as a form of template that we must instantiate separately.
- Uninstantiated packages and package instances that are declared as design units and stored in a design library, they can be written as a further form of design unit

4.10 Conformance Rules

- Two specifications for one subprogram are conform if
  - A numeric literal can be replaced by a different numeric literal if and only if both have the same value.

– A simple name can be replaced by an expanded name in which this simple name is the suffix if, and only if, at both places the meaning of the simple name is given by the same declaration.

both specifications are made up of the same sequence of lexical elements and corresponding lexical elements have the same meaning