Mixed Netlists

Ernst Christen (Ernst.Christen@synopsys.com)
John Shields (jshields@acm.org)

1. Introduction

This paper defines syntax and semantics to extend IEEE Std. 1076.1-1999 to support mixed nets, i.e. nets that are not purely nodes, or quantity nets, or event-driven nets, but in which objects of different kinds participate. Mixed nets arise naturally when using schematic-based design entry, by connecting, in the schematic, ports of unlike kinds by means of a wire. The definitions are based on the concepts of a structural wire and views of the wire. Each view provides the semantics of an object of a particular kind and type or nature, and it can be configured to be implemented by an instance of a design unit that converts between the wire and the object.

1.1 Requirements

The DOD contains a number of objectives related to the interface between a continuous object (a terminal or a quantity) and a discrete object (a signal). We believe that the same kind of objectives should apply to the interface between a terminal and a quantity.

In many cases quantity ports and signal ports are abstractions of conservative connections, which in the 1076.1 language are described by terminal ports with appropriate reference terminals. The interface between different objects must therefore bridge different levels of abstraction, which requires the capability to describe arbitrary behavior at the interface. The following list of requirements for mixed netlists takes this into consideration.

a) Support top-down design, by supporting the hierarchical decomposition of a model that includes interfaces between signals, terminals, and quantities.

b) Support unidirectional interfaces (e.g. quantity -> terminal, terminal -> quantity)

c) Support bidirectional interfaces (e.g. for a bus driver/receiver)

d) Support modeling of such effects as input/output impedance, timing, current drive capability.

e) DO12: The model of the interface between analog and digital descriptions should be customizable by the user. We extend this to also include the interface between terminals and quantities.

f) DO32: Default conversions between analog and digital connection points should be provided. We extend this to also include the interface between terminals and quantities.

g) Support "analog" and "digital" implementations of a design unit that can be selected by configurations.

2. Definitions

The definitions are written in the form of text to be inserted into the VHDL 1076.1-1999 LRM. Paragraphs enclosed in brackets set the context and are not part of the definition itself. A reference of the form LRM 4.3.1-53.3 is to the IEEE 1076.1-1999 Language Reference Manual, Section 4.3.1, 3rd paragraph on p.53.

2.1 Shapes

[at LRM 3.6?]

3.6 Shapes

A *shape* defines the properties of a wire, expressed by referring to the structural properties of a type or nature by an attribute name. The shape of a scalar type or a scalar nature is a scalar. The shape of a record type or a record nature is a record in which there is a matching element for each element of the record type or record nature whose name is the name of the element of the record type or record nature and whose shape is the shape of the element of the record type or record nature. The shape of an unconstrained array type is an unconstrained array whose elements have the shape of the subtype defined by the element subtype indication of the unconstrained array definition and whose index subtype definitions are the index subtype definitions of the unconstrained array type. Similarly, the shape of an unconstrained array nature is an unconstrained array whose elements have the shape of the subnature defined by the subnature indication of the unconstrained nature definition and whose index subtype definitions are the index subtype definitions of the unconstrained array nature. The shape of an constrained array type is a constrained array whose elements have the shape of the subtype defined by the element subtype indication of the constrained array definition and whose index constraints are the index constraints of the constrained array definition. Similarly, the shape of an constrained array nature is a constrained array whose elements have the shape of the subnature defined by the subnature indication of the constrained nature definition and whose index constraints are the index constraints of the constrained nature definition.

Note—It is a consequence of these rules that a nature and the across and through types defined by the nature definition have the same shape.

[at 14.1]

T’SHAPE

Kind:	Shape.
Prefix:	Any type denoted by the static name T.
Result:	The shape of the type denoted by T.

N’SHAPE

Kind:	Shape.
Prefix:	Any nature denoted by the static name N.

Result: The shape of the nature denoted by N.

2.2 Wires

[at 4.3.1]

An object declaration declares an object of a specified type, nature, or shape. Such an object is called an *explicitly declared object*.

object_declaration ::=
    ...
    | wire_declaration

[at 4.3.1.7]

4.3.1.7 Wire declarations

A wire declaration declares one or more wires. A wire is an object that during elaboration is converted to a signal or a quantity of a particular type with a particular mode (if applicable), or to a terminal with a particular nature. The shape of the converted object matches the shape of the wire.

wire_declaration ::=
    wire identifier_list : shape_indication ;

shape_indication ::=
      type_mark ' SHAPE
    | nature_mark ' SHAPE

Each wire of a scalar shape and each scalar subelement of a composite shape is a *scalar wire*.

[at 4.3.2]

An interface declaration declares an *interface object* of a specified type, nature, or shape. ...; *interface terminals*, i*nterface quantities*, and *interface wires* that appear as ports of a design entity, a component, or a block.

interface_declaration ::=
    ...
    | interface_wire_declaration

interface_wire_declaration ::=
    wire identifier_list : shape_indication

NOTES

4—An interface object of class wire gets a mode during elaboration if the wire is converted to a signal or quantity.

[at 4.3.2.1-64.1]

add interface wire declaration

[at 4.3.2.2]

actual_designator ::=
...
| *wire*_name
| **open**

[at 4.3.2.2-66.2]

[Need text defining legal forms of association elements]

[There are additional places in the LRM where objects or interface objects are enumerated. These must be identified and updated]

2.3 Wire Views

[at 4.3.2.3]

4.3.2.3 Wire views

The name of a wire can appear in the text of a model only in a wire declaration, an interface wire declaration, in a wire object specification, and as the prefix of an attribute name that defines a view of the wire. In all other places a wire must be referenced through a *wire view* that defines the object kind, type or nature, and mode (if applicable) of the reference.

A wire view is specified by an attribute name whose prefix is a wire name and whose arguments specify the nature or type and mode (if applicable) of the wire view. In the text of a model, a wire view of kind terminal with a specified nature can appear anywhere the name of a terminal of the nature can appear. Similarly, a wire view of kind quantity with a specified type and mode can appear anywhere the name of a quantity of the type and mode can appear. Finally, a wire viewof kind signal with a specified type and mode can appear anywhere the name of a signal of the type and mode can appear.

It is an error if more than one wire view for the same wire appears in a declarative region. [This is not completely correct if the same wire is associated as an actual with more than one formal port in a port map. More work is required.]

Examples:

```
library ieee;
use ieee.electrical_systems.all;
entity inverter is
    port (wire input, output: REAL'SHAPE; terminal supply: electrical);
end entity inverter;
```

```
architecture digital of inverter is
begin
    output'SIGNAL(BIT, out) <= not input'SIGNAL(BIT, in);
end architecture digital;

architecture analog of inverter is
    quantity vin across input'TERMINAL(electrical);
    quantity vout across iout through output'TERMINAL(electrical);
    quantity vcc across supply;
begin
    vout == vcc - vin;
end architecture analog;
```

[at 14.1]

W'TERMINAL(N)

Kind:	Terminal.
Prefix:	Any wire denoted by the static name W.
Parameter:	A nature mark denoted by the name N.
Result nature:	The nature defined by the nature mark N.
Result:	A terminal whose nature is N.
Restrictions:	N'SHAPE must match the shape of W.

W'QUANTITY(T, mode)

Kind:	Quantity.
Prefix:	Any wire denoted by the static name W.
Parameters:	T: A type mark denoted by the name T. mode: The mode specifying how the quantity view is used. Must be either **in** or **out**.
Result type:	The type defined by the type mark T.
Result:	A quantity whose type is T and whose mode is as specified.
Restrictions:	T'SHAPE must match the shape of W.

W'SIGNAL(T, mode)

Kind:	Signal.
Prefix:	Any wire denoted by the static name W.
Parameters:	T: A type mark denoted by the name T. mode: The mode specifying how the signal view is used. Must be **in**, **out**, **inout**, or **buffer**.
Result type:	The type defined by the type mark T.
Result:	A signal whose type is T and whose mode is as specified.
Restrictions:	T'SHAPE must match the shape of W.

2.4 Wire Configurations

[at 5.5?]

5.5 Wire configuration specification

A wire configuration specification identifies a collection or a class of wires and associates binding information with the wire views of these wires. The wires may appear in the port association list or the declarative region of the block in which the wire configuration specification appears and any block nested within the block.

wire_configuration_specification ::=
for *wire*_object_specification
{ conversion_specification }
end for ;
object_specification ::=
terminal name_list : nature_mark
| **quantity** name_list : [**in** | **out**] type_mark
| **signal** name_list : [mode] type_mark
name_list ::=
simple_name { , simple_name }
| **others**
| **all**

The wire object specification identifies the wires with whose wire views binding information is to be associated, as follows:

- If the reserved word **terminal**, a list of simple names and a nature mark is supplied, then the wire configuration specification applies to the wire views of the wires denoted by the simple names that have been converted during elaboration to terminals whose nature is the nature denoted by the nature mark.
- If the reserved words **terminal** and **others** and a nature mark is supplied, then the wire configuration specification applies to the wire views of the wires that have been converted during elaboration to terminals whose nature is the nature denoted by the nature mark, provided that each such wire is not explicitly named in the name list of a previous wire object specification with the reserved word **terminal** and the same nature mark.
- If the reserved words **terminal** and **all** and a nature mark is supplied, then the wire configuration specification applies to the wire views of all wires that have been converted during elaboration to terminals whose nature is the nature denoted by the nature mark.
- If the reserved word **quantity**, a list of simple names, a type mark and optionally a mode is supplied, then the wire configuration specification applies to the wire views of the wires denoted by the simple names that have been converted during elaboration to quantities whose type is the type denoted by the type mark and whose mode, if applicable, is the specified mode.

- If the reserved words **quantity** and **others**, a type mark and optionally a mode is supplied, then the wire configuration specification applies to the wire views of the wires that have been converted during elaboration to quantities whose type is the type denoted by the type mark and whose mode, if applicable, is the specified mode, provided that each such wire is not explicitly named in the name list of a previous wire object specification with the reserved word **quantity** and the same type mark and mode, if applicable.
- If the reserved words **quantity** and **all**, a type mark and optionally a mode is supplied, then the wire configuration specification applies to the wire views of all wires that have been converted during elaboration to quantities whose type is the type denoted by the type mark and whose mode, if applicable, is the specified mode.
- If the reserved word **signal**, a list of simple names, a type mark and optionally a mode is supplied, then the wire configuration specification applies to the wire views of the wires whose name denoted by the simple names that have been converted during elaboration to signals whose type is the type denoted by the type mark and whose mode, if applicable, is the specified mode.
- If the reserved words **aignal** and **others**, a type mark and optionally a mode is supplied, then the wire configuration specification applies to the wire views of the wires that have been converted during elaboration to signals whose type is the type denoted by the type mark and whose mode, if applicable, is the specified mode, provided that each such wire is not explicitly named in the name list of a previous wire object specification with the reserved word **signal** and the same type mark and mode, if applicable.
- If the reserved words **signal** and **all**, a type mark and optionally a mode is supplied, then the wire configuration specification applies to the wire views of all wires that have been converted during elaboration to signals whose type is the type denoted by the type mark and whose mode, if applicable, is the specified mode.

5.5.1 Conversion specification

A conversion specification associates binding information with wire views of the wires identified by the enclosing wire configuration specification.

conversion_specification ::=
for object_specification binding_indication ;

The object specification further specifies the wire view with which binding information is to be associated, as follows:

- If a list of simple names is supplied, then each simple name must denote a terminal port, a quantity port, or a signal port, and the conversion specification applies to the wire view associated as an actual with the port as a formal.
- If the reserved word **terminal**, a list of simple names and a nature mark is supplied, then the conversion specification applies to the wire views whose kind is terminal and whose nature is the nature denoted by the nature mark, except the wire views associated as actuals with a formal port whose name appears in an object name list in the wire configuration specification.

- If the reserved word **quantity**, a type mark and optionally a mode is supplied, then the conversion specification applies to the wire views whose kind is quantity, whose type is the type denoted by the type mark and whose mode, if applicable, is the mode specified by the reserved words **in** or **out**, except the wire views associated as actuals with a formal port whose name appears in an object name list in the wire configuration specification.
- If the reserved word **signal**, a type mark and optionally a mode is supplied, then the conversion specification applies to the wire views whose kind is signal, whose type is the type denoted by the type mark and whose mode, if applicable, is the specified mode, except the wire views associated as actuals with a formal port whose name appears in an object name list in the wire configuration specification.

It is an error if the object specification of a conversion specification is a name list and the wire object specification of the enclosing wire configuration specification is also a name list. Similarly, it is an error if the object specification of a conversion specification includes one of the reserved word **terminal**, **quantity**, or **signal** and the wire object specification of the enclosing wire configuration specification includes the same reserved word. It is also an error if the object specification of a conversion specification is a name list or includes a mode and the wire object specification of the enclosing wire configuration specification includes a mode. Conversely, it is an error if the object specification of a conversion specification that includes the reserved words **quantity** or **signal** does not include a mode and the wire object specification of the enclosing wire configuration specification also does not include a mode.

2.5 Elaboration

[at 12.1.1]

12.1.1 Wire Conversion

A *wire root* is a wire declared by a wire declaration or a wire port associated with an actual that designates a terminal, quantity, or signal. A *wire net* is a wire root and the transitive closure of each wire port that is associated with an actual that is either the wire root or a wire port of the wire net. The *members* of a wire net are the wire root and each wire port of the wire net.

After the elaboration steps described in 12.1 have been completed, each wire net is converted as follows:

- If a member of a wire net has a wire view that is a terminal, then the wire declaration defining the wire root, if any, is converted to a terminal declaration whose nature is the nature of the wire view, and each wire port that is a member of the wire net is converted to a terminal port whose nature is the nature of the wire view. It is an error if the wire net has more than one wire view of kind terminal and the natures of these wire views don’t match.

- Otherwise, if a member of a wire net has a wire view that is a quantity, then the wire declaration defining the wire root, if any, is converted to a quantity declaration whose type is the type of the wire view, and each wire port that is a member of the wire net is converted to a quantity port whose type is the type of the wire view. It is an error if the wire net has more than one wire view of kind quantity and the types of these wire views don't match.

  The mode of each quantity port (including the quantity port at the root of the net, if any) is determined as follows, The *contributing modes* of a port are the mode of each wire view, if any, of the wire port being converted to the port, and the mode of each (possibly converted) formal port, if any, whose associated actual is the port. If all contributing modes of the quantity port are **in**, then the mode of the quantity port is **in**. If exactly one of the contributing modes of the quantity port is **out**, **inout**, or **buffer**, then the mode of the quantity port is **out**. It is an error if more than one contributing mode of a quantity port is **out**, **inout**, or **buffer**.

- Otherwise, if a member of a wire net has a wire view that is a signal, then the wire declaration defining the wire root, if any, is converted to a signal declaration whose type is the type of the wire view, and each wire port that is a member of the wire net is converted to a signal port whose type is the type of the wire view. It is an error if the wire net has more than one wire view of kind signal and the types of these wire views don't match.

  The mode of each signal port (including the signal port at the root of the net, if any) is determined as follows, If all the contributing modes of a signal port are the same, then the mode of the signal port is this mode, otherwise the mode of the signal port is **inout**.

It is an error if a wire net has no wire view.

[at 12.2.4-159.2]

Elaboration of a port association list consists of the elaboration of each port association element in the association list whose actual is not the reserved word **open**. Elaboration of a port association element consists of the elaboration of the formal part followed by the association of the port or subelement or slice thereof with the object or subelement or slice thereof designated by the actual part. If the formal part is a wire port or subelement or slice thereof and the actual part does not designate a wire, the following rules apply:

- If the actual part designates a terminal, then the formal part is replaced by a wire view whose kind is terminal and whose nature is the nature of the actual part.
- If the actual part designates a quantity, then the formal part is replaced by a wire view whose kind is quantity and whose type is the type of the actual part. The mode of the wire view is left unspecified.
- If the actual part designates a signal, then the formal part is replaced by a wire view whose kind is signal and whose type is the type of the actual part. The mode of the wire view is left unspecified.

Similarly, if the actual part of a port association element designates a wire and the formal part is not a wire port or subelement or slice thereof, the following rules apply:

- If the formal part is a terminal port or subelement or slice thereof, then the actual part is replaced by a wire view whose kind is terminal and whose nature is the nature of the formal part.
- If the formal part is a quantity port or subelement or slice thereof, then the actual part is replaced by a wire view whose kind is quantity and whose type and mode is the type and mode, respectively, of the formal part.
- If the formal part is a signal port or subelement or slice thereof, then the actual part is replaced by a wire view whose kind is signal and whose type and mode is the type and mode, respectively, of the formal part.

If the elaboration of a port association element does not yield a formal part with an unspecified mode, then the association of the formal part with the actual part also involves a check that the restrictions on port associations (see 1.1.1.2) are met. It is an error if this check fails.

[Needs work to handle individual subelement associations]

[Additional work in the remainder of this section to accommodate wire views]

3. References

[DOD] 1076.1 Design Objective Document V2.3

Revision History

0.1 Sep.8, 1995 Ernst Christen
Original version

0.2 Jan.14, 1996 Ernst Christen
Major update, including comments from LDC meetings and information about simple mixed associations.

0.3 April 24, 1996 Ernst Christen
Replaced VHDL-A by VHDL 1076.1 or equivalent.

0.4 July 25, 2004 Ernst Christen, John Shields
Complete rework, based on structural wires.

0.5 June 19, 2005 Ernst Christen, John Shields
Updated based on work on paper for FDL'04