1. Introduction

EDITORS NOTE: 10th May 2011. I addressed the points that Shalom raised in the 31st March meeting. Item 5 - I did not change, “analog” is a keyword, “procedural block” is plain text. This is intended.

1.1 Overview

The SystemVerilog-AMS Hardware Description Verification Language (HDVL) language reference manual defines a behavioraHDVLl language for analog and mixed-signal systems. SystemVerilog-AMS HDVL is derived from IEEE Std 1800-2009 SystemVerilog HDVL. This document is intended to cover the definition and semantics of SystemVerilog-AMS HDVL as proposed by Accellera. Accellera is a consortium of EDA, semiconductor, and system companies.

SystemVerilog-AMS HDVL consists of the complete IEEE Std 1800-2009 SystemVerilog HDVL specification, as well as the Verilog-AMS 2.3.1 HDVL, and extensions to both for specifying the full SystemVerilog-AMS HDVL.

SystemVerilog-AMS HDVL lets designers of analog and mixed-signal systems and integrated circuits create and use modules which encapsulate high-level behavioral descriptions as well as structural descriptions of systems and components. The behavior of each module can be described mathematically in terms of its ports and external parameters applied to the module. The structure of each component can be described in terms of interconnected sub-components. These descriptions can be used in many disciplines such as electrical, mechanical, fluid dynamics, and thermodynamics.

For continuous systems, SystemVerilog-AMS HDVL is defined to be applicable to both electrical and non-electrical systems description. It supports *conservative* and *signal-flow* descriptions by using the concepts of *nets, nodes*, *branches*, and *ports* as terminology for these descriptions. The solution of analog behaviors which obey the laws of conservation fall within the generalized form of Kirchhoff’s Potential and Flow Laws (KPL and KFL). Both of these are defined in terms of the quantities (e.g., voltage and current) associated with the analog behaviors.

SystemVerilog-AMS HDVL can also be used to describe discrete (digital) systems (per IEEE Std 1800-2009 SystemVerilog HDVL) and mixed-signal systems using both discrete and continuous descriptions as defined in this LRM.

1.2 Mixed-signal language features

SystemVerilog-AMS HDVL extends the features of the digital modeling language (IEEE Std 1800-2009 SystemVerilog HDVL) to provide a single unified language with both analog and digital semantics with backward compatibility. Below is a list of salient features of the resulting language:

- signals of both analog and digital types can be declared in the same module
- **`initial`**, **`always`**, and **`analog`** procedural blocks can appear in the same module
- both analog and digital signal values can be accessed (read operations) from any context (analog or digital) in the same module
- digital signal values can be set (write operations) from any context outside of an `analog` procedural block
- analog potentials and flows can only receive contributions (write operations) from inside an analog procedural block
- the semantics of the **`initial`** and **`always`** blocks remain the same as in IEEE Std 1800-2009 SystemVerilog HDVL; the semantics for the **`analog`** block are described in this manual
- the **`discipline`** declaration is extended to digital signals

Copyright © 2011 Accellera Organization, Inc. All rights reserved.

— a new construct, **`connect`** statement, is added to facilitate auto-insertion of user-defined connection modules between the analog and digital domains

— when hierarchical connections are of mixed type (i.e., analog signal connected to digital port or digital signal connected to analog port), user-defined connection modules are automatically inserted to perform signal value conversion

1.3 Systems

A *system* is considered to be a collection of interconnected *components* which are acted upon by a stimulus and produce a response. The components themselves can also be systems, in which case a *hierarchical system* is defined. If a component does not have any subcomponents, it is considered to be a *primitive component*. Each primitive component connects to zero or more nets. Each net connects to a signal which can traverse multiple levels of the hierarchy. The behavior of each component is defined in terms of values at each net.

A *signal* is a hierarchical collection of nets which, because of port connections, are contiguous. If all the nets which make up a signal are in the discrete domain, the signal is a *digital signal*. If, on the other hand, all the nets which make up a signal are in the continuous domain, the signal is an *analog signal*. A signal which consists of nets from both domains is called a *mixed signal*.

Similarly, a port whose connections are both analog is an *analog port*, a port whose connections are both digital is a *digital port*, and a port whose connections are both analog and digital is a *mixed port*. The components connect to nodes through ports and nets to build a hierarchy, as shown in Figure 1-1.

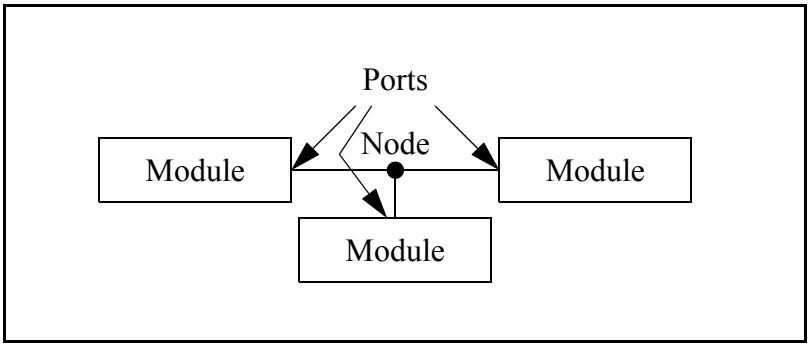

Figure 1-1: Components connect to nodes through ports

If a signal is analog or mixed, it is associated with a node (see 3.6), while a purely digital signal is not associated with a node. Regardless of the number of analog nets in an analog or mixed signal or how the analog nets in a mixed signal are interspersed with digital nets, the analog portion of an analog or mixed signal is represented by only a single node. This guarantees a mixed or analog signal has only one value which represents its potential with respect to the global reference voltage (*ground*).

In order to simulate systems, it is necessary to have a complete description of the system and all of its components. Descriptions of systems are usually given structurally. That is, the description of a system contains instances of components and how they are interconnected. Descriptions of components are given using behavior and or structure. A behavior is a mathematical description which relates the signals at the ports of the components.

Copyright © 2011 Accellera Organization, Inc.

1.3.1 Conservative systems

An important characteristic of conservative systems is that there are two values associated with every node, the *potential* (also known as the *across value* or *voltage* in electrical systems) and the *flow* (the *through value* or *current* in electrical systems). The potential of the node is shared with all continuous ports and nets connected to the node so all continuous ports and nets see the same potential. The flow is shared so flow from all continuous ports and nets at a node shall sum to zero (`0`). In this way, the node acts as an infinitesimal point of interconnection in which the potential is the same everywhere on the node and on which no flow can accumulate. Thus, the node embodies Kirchhoff's Potential and Flow Laws (KPL and KFL). When a component connects to a node through a conservative port or net, it can either affect, or be affected by, either the potential at the node, and/or the flow onto the node through the port or net.

With conservative systems it is also useful to define the concept of a branch. A branch is a path of flow between two nodes through a component. Every branch has an associated potential (the potential difference between the two nodes) and flow.

A behavioral description of a conservative component is constructed as a collection of interconnected branches. The constitutive equations of the component are formulated as to relate the branch potentials and flows. In the probe/source approach (see 5.4.2), the branch potential or flow is specified as a function of branch potentials and flows. If the branch potential and flow are left unspecified, not on the left-hand side of a contribution statement, then the branch acts as a probe. In this case, if the branch flow is used in an expression, the branch potential is forced to zero (`0`). Otherwise the branch flow is assumed to be zero (`0`) and the branch potential is available for use in an expression. Using both the potential and flow of a 'probe' branch in an expression is not allowed. Nor is specifying both the branch potential and flow at the same time. (While these last two conditions are not really necessary, they do eliminate conditions which are useless and confusing.)

1.3.1.1 Reference nodes

The potential of a single node is given with respect to a reference node. The potential of the reference node, which is called **`ground`** in electrical systems, is always zero (`0`). Any net of continuous discipline can be declared to be **`ground`**. In this case, the node associated with the net shall be the global reference node in the circuit. This is compatible with all analog disciplines and can be used to bind a port of an instantiated module to the reference node.

1.3.1.2 Reference directions

The reference directions for a generic branch are shown in Figure 1-2.

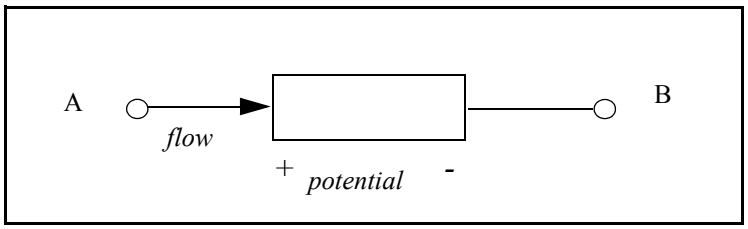

Figure 1-2: Reference directions

The *reference direction* for a potential is indicated by the plus and minus symbols near each port. Given the chosen reference direction, the branch potential is positive whenever the potential of the port marked with a plus sign (`A`) is larger than the potential of the port marked with a minus sign (`B`). Similarly, the flow is positive whenever it moves in the direction of the arrow (in this case from `+` to `-`).

Copyright © 2011 Accellera Organization, Inc. All rights reserved.

SystemVerilog-AMS HDVL uses associated reference directions. A positive flow enters a branch through the port marked with the plus sign and exits the branch through the port marked with the minus sign.

1.3.2 Kirchhoff's Laws

In formulating continuous system equations, SystemVerilog-AMS HDVL uses two sets of relationships. The first are the constitutive relationships which describe the behavior of each component. Constitutive relationships can be kept inside the simulator as built-in primitives or they can be provided by SystemVerilog-AMS HDVL module definitions.

The second set of relationships, interconnection relationships, describe the structure of the network. Interconnection relationships, which contain information on how the components are connected to each other, are only a function of the system topology. They are independent of the nature of the components.

A SystemVerilog-AMS HDVL simulator uses Kirchhoff's Laws to define the relationships between the nodes and the branches. Kirchhoff's Laws are typically associated with electrical circuits that relate voltages and currents. However, by generalizing the concepts of voltages and currents to potentials and flows, Kirchhoff's Laws can be used to formulate interconnection relationships for any type of system.

Kirchhoff's Laws provide the following properties relating the quantities present on nodes and branches, as shown in Figure 1-3.

— Kirchhoff's Flow Law (KFL)
The algebraic sum of all flows out of a node at any instant is zero (0).

— Kirchhoff's Potential Law (KPL)
The algebraic sum of all the branch potentials around a loop at any instant is zero (0).

These laws imply a node is infinitely small; so there is negligible difference in potential between any two points on the node and a negligible accumulation of flow.

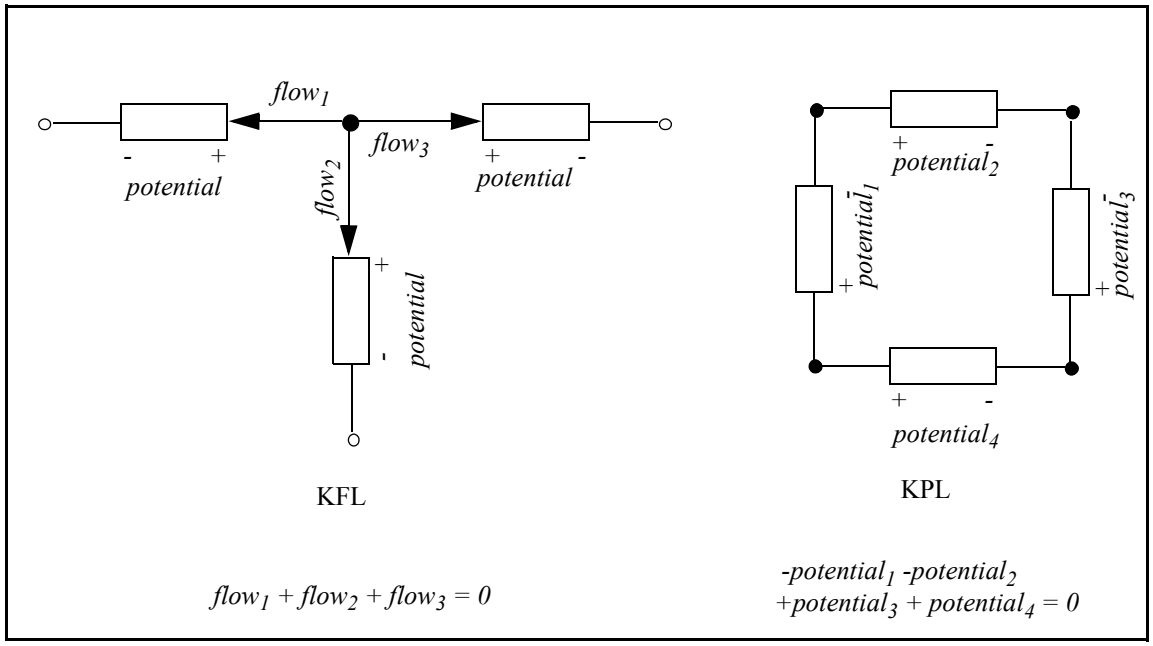

Figure 1-3: Kirchhoff's Flow Law (KFL) and Potential Law (KPL)

1.3.3 Natures, disciplines, and nets

SystemVerilog-AMS HDVL allows definition of nets based on disciplines. The disciplines associate potential and flow natures for conservative systems or only potential nature for signal-flow systems. The natures are a collection of attributes, including user-defined attributes, which describes the units (meter, gram, newton, etc.), absolute tolerance for convergence, and the names of potential and flow access functions.

The disciplines and natures can be shared by many nets. The compatibility rules help enforce the legal operations between nets of different disciplines.

1.3.4 Signal-flow systems

A discipline may specify two nature bindings, **`potential`** and **`flow`**, or it may specify only a single binding, **`potential`**. Disciplines with two natures are know as *conservative disciplines* because nodes which are bound to them exhibit Kirchhoff's Flow Law, and thus, conserve charge (in the electrical case). A discipline with only a potential nature is known as a *signal flow discipline*.

As a result of port connections of analog nets, a single node may be bound to a number of nets of different disciplines. If a node is bound only to disciplines which have potential nature only, current contributions to that node are not legal. Flow for such a node is not defined.

Signal flow models may be written so potentials of module outputs are purely functions of potentials at the inputs without taking flow into account.

The following example is a level shifting voltage follower:

```
module shiftPlus5(in, out);
input in;
output out;
voltage in, out;  //voltage is a signal flow
                  //discipline compatible with
                  //electrical, but having a
                  //potential nature only
analog begin
   V(out) <+ 5.0 + V(in);
end
endmodule
```

If a number of such modules were cascaded in series, it would not be necessary to conserve charge (i.e., sum the flows) at any intervening node.

If, on the other hand, the output of this device were bound to a node of a conservative discipline (e.g., `electrical`), then the output of the device would appear to be a controlled voltage source to ground at that node. In that case, the flow (i.e., current) through the source would contribute to charge conservation at the node. If the input of this device were bound to a node of a conservative discipline then the input would act as a voltage probe to ground. Thus, when a net of signal flow discipline with potential nature only is bound to a conservative node, contributions made to that net behave as voltage sources to ground.

Nets of signal flow disciplines in modules may only be bound to **`input`** or **`output`** ports of the module, not to **`inout`** ports. Contributions may not be made to **`input`** ports.

1.3.5 Mixed conservative/signal flow systems

When practicing the top-down design style, it is extremely useful to mix conservative and signal-flow components in the same system. Users typically use signal-flow models early in the design cycle when the sys-

Copyright © 2011 Accellera Organization, Inc. All rights reserved.

tem is described in abstract terms, and gradually convert component models to conservative form as the design progresses. Thus, it is important to be able to initially describe a component using a signal-flow model, and later convert it to a conservative model, with minimum changes. It is also important to allow conservative and signal-flow components to be arbitrarily mixed in the same system.

The approach taken is to write component descriptions using conservative semantics, except port and net declarations only require types for those values which are actually used in the description. Thus, signal-flow ports only require the type of potential to be specified, whereas conservative ports require types for both values (the potential and flow).

For example, consider a differential voltage amplifier, a differential current amplifier, and a resistor. The amplifiers are written using signal-flow ports and the resistor uses conservative ports.

```
module voltage_amplifier (out, in);
input in;
output out;
voltage  out,     // Discipline voltage defined elsewhere
         in;      // with access function V()
parameter real GAIN_V = 10.0;

analog
   V(out) <+ GAIN_V * V(in);

endmodule
```

In this case, only the voltage on the ports are declared because only voltage is used in the body of the model.

```
module current_amplifier (out, in);
input in;
output out;
current  out,     // Discipline current defined elsewhere
         in;      // with access function I()
parameter real GAIN_I = 10.0;

analog
   I(out) <+ GAIN_I * I(in);

endmodule
```

Here, only current is used in the body of the model, so only current need be declared at the ports.

```
module resistor (a, b);
inout a, b;
electrical a, b;        // access functions are V() and I()
parameter real R = 1.0;

analog
   V(a,b) <+ R * I(a,b);

endmodule
```

The description of the resistor relates both the voltage and current on the ports. Both are defined in the conservative discipline `electrical`.

Copyright © 2011 Accellera Organization, Inc.

In summary, only those signals types declared on the ports are accessible in the body of the model. Conversely, only those signals types used in the body need be declared.

This approach provides all of the power of the conservative formulation for both signal-flow and conservative ports, without forcing types to be declared for unused signals on signal-flow nets and ports. In this way, the first benefit of the traditional signal-flow formulation is provided without the restrictions.

The second benefit, that of a smaller, more efficient set of equations to solve, is provided in a manner which is hidden from the user. The simulator begins by treating all ports as being conservative, which allows the connection of signal-flow and conservative ports. This results in additional unnecessary equations for those nodes which only have signal-flow ports. This situation can be recognized by the simulator and those equations eliminated.

Thus, this approach to allowing mixed conservative/signal-flow descriptions provides the following benefits:

— Conservative components and signal-flow components can be freely mixed. In addition, signal-flow components can be converted to conservative components, and vice versa, by modifying only the component behavioral description.
— Many of the capabilities of conservative ports, such as the ability to access flow and the ability to access floating potentials, are available with signal-flow ports.
— Natures only have to be given for potentials and flows if they are accessed in a behavioral description.
— If nets and ports are used only in a structural description (only in instance statements), then no natures need be specified.

1.4 Conventions used in this document

This document is organized into sections, each of which focuses on some specific area of the language. There are subsections within each section to discuss individual constructs and concepts. The discussion begins with an introduction and an optional rationale for the construct or the concept, followed by syntax and semantic description, followed by some examples and notes.

The formal syntax of SystemVerilog-AMS HDVL is described using Backus-Naur Form (BNF). The following conventions are used:

1) Lower case words, some containing embedded underscores, are used to denote syntactic categories. For example:

   module_declaration

2) Boldface red characters denote reserved keywords, operators and punctuation marks as required part of the syntax. For example:

```
module   =   ;
```

3) Blue characters are used to denote syntax productions that are SystemVerilog-AMS extensions to IEEE Std 1800-2009 SystemVerilog HDVL syntax. For example:

```
connectrules_declaration ::=
  connectrules connectrules_identifier ;
    { connectrules_item }
  endconnectrules
```

4) A vertical bar (|) that is not in boldface-red separates alternative items. For example:

Copyright © 2011 Accellera Organization, Inc. All rights reserved.

attribute ::=
abstol | **units** | identifier

5) Square brackets ([ ]) that are not in boldface-red enclose optional items. For example:

input_declaration ::=
input [range] list_of_ports **;**

6) Braces ({ }) that are not in boldface-red enclose a repeated item unless the braces appear in bold face, in which case it stands for itself. The item can appear zero or more times; the repetitions occur from left to right as with an equivalent left-recursive rule. Thus, the following two rules are equivalent:

list_of_port_def ::=
port_def { **,** port_def }

list_of_port_def ::=
port_def
| list_of_port_def **,** port_def

7) A *qualified term* in the syntax is a term such as *array_identifier* for which the "array" portion represents some semantic intent and the "identifier" term indicates that the qualified term reduces to the "identifier" term in the syntax. The syntax does not completely define the semantics of such qualified terms; for example while an identifier which would qualify semantically as an array_identifier is created by a declaration, such declaration forms are not explicitly described using *array_identifier* in the syntax.

The main text uses italicized font when a *term* is being defined, and constant-width font for `examples`, `file names`, and while referring to `constants`. Reserved keywords in the main text and in examples are in a **`constant-width bold`** font.

1.5 Contents

This document contains the following clauses and annexes:

Copyright © 2011 Accellera Organization, Inc.

6. Hierarchical structures
This clause describes how to build hierarchical descriptions using SystemVerilog-AMS HDVL.

7. Mixed signal
This clause describes the mixed-signal aspects of the SystemVerilog-AMS HDVL language.

8. Scheduling semantics
This clause describes the basic simulation cycle as applicable to SystemVerilog-AMS HDVL.

9. System tasks and functions
This clause describes the system tasks and functions in SystemVerilog-AMS HDVL.

10. Compiler directives
This clause describes the compiler directives in SystemVerilog-AMS HDVL.

11. Using VPI routines
This clause describes how the VPI routines are used.

12. VPI routine definitions
This clause defines each of the VPI routines in alphabetical order.

A. (normative) Formal syntax definition
This annex describes formal syntax for all SystemVerilog-AMS HDVL constructs in Backus-Naur Form (BNF).

B. (normative) List of keywords
This annex lists all the words which are recognized in SystemVerilog-AMS HDVL as keywords.

C. (normative) Analog language subset
This annex describes the analog subset of SystemVerilog-AMS HDVL.

D. (normative) Standard definitions
This annex provides the definitions of several natures, disciplines, and constants which are useful for writing models in SystemVerilog-AMS HDVL.

E. (normative) SPICE compatibility
This annex describes the Spice compatibility with SystemVerilog-AMS HDVL.

F. (normative) Discipline resolution methods
This annex provides the semantics for two methods of resolving the discipline of undeclared interconnect.

G. (informative) Change history
This annex provides a list of changes between various versions of the SystemVerilog-AMS Language Reference Manual.

H. (informative) Glossary
This annex describes various terms used in this document.

Copyright © 2011 Accellera Organization, Inc. All rights reserved.