IEEE VHDL subPAR 1076.1

Analog Extensions to VHDL

Design Objective Document (DOD)

Version 1.1
March 2, 1993

Document history

1.1 2 mar 93 Alain Vachoux
Add new appendix A with consolidated list of submitted requirements,
prepared by Mart Altmae and Kevin Nolan.

1.0 30 nov 92 Alain Vachoux
Revised following meetings at DAC (Anaheim, June 92), EuroDAC (Hamburg, September 92) and IEEE DASS (Washington DC, October 1992).
New document structure to highlight and to number the design objectives.

0.2 11 may 92 Robert Cottrell
Revised following meeting at European VHDL Forum in Santander, Spain, 28 apr 92.

0.1 2 apr 92 Hazem El Tahawy
Initial version prepared by Bergé, El Tahawy, Rodriguez and Rouquier.

Table of contents

1. Introduction 1
2. Design requirements 2
3. General guidelines 3
   3.1. Scope of the language 3
   3.2. Digital aspects 3
   3.3. Analog aspects 3
   3.4. Mixed aspects 4
4. Analog structures 4
   4.1. Similarities with the digital domain 4
   4.2. Specific aspects of analog domain 5
5. Analog behaviors 6
   5.1. Relations 6
   5.2. Procedural models 6
   5.3. Access to voltages and currents 6
6. Interactions between analog and digital domains 7
   6.1. Analog to digital interactions 7
   6.2. Digital to analog interactions 8
7. Units 8
   7.1. Entities 8
   7.2. Architectures 8
   7.3. Math package 9
   7.4. Electrical package 9
   7.5. Dimensional analysis package 9
8. Simulation mechanisms 10
   8.1. Time management 10
   8.2. Initialisation 10
   8.3. Simulation control 10
9. Open questions 11
   9.1. Relations 11
   9.2. Procedural models 11
   9.3. Modification to the digital simulation kernel 11
   9.4. Assumptions on the analog simulator 12
   9.5. Miscellaneous questions 12

Appendix A: Detailed analysis of design requirements 13

Appendix B: Summary of submitted requirements 18

1. Introduction

The Design Objective Document (DOD) is a working document that aims to serve as the basis on which the design of analog extensions to VHDL is performed. This work is done within the subPAR 1076.1 standardisation activities and it will produce a separate VHDL norm, called VHDL-A, until the merge with VHDL 92 in the next standardisation of 1997.

The DOD is build from a list of design requirements submitted to the subPAR 1076.1 for analog extensions to VHDL. The complete list of these requirements was collected by Mark Altmae and Kevin Nolan from separate European and North American requirements. The list of relevant requirements, further called ***Design Requirements*** (DR) in this document, is recalled below. Design requirements should be met by VHDL-A and are the focus of the language design activity.

The DOD contains a list of numbered ***Design Objectives*** (DO) along with some comments to explain or to highlight specific aspects of them. A design objective has to be studied and possibly implemented in order to verify or improve its functionality.

The DOD contains as few syntactic examples as possible. Some ideas are explicitly supported by one or several design requirements. The corresponding requirement numbers are listed after the corresponding design objective. Other ideas are not explicitly supported by any requirement, but appeared during analysis, or during discussions.

The definitions of VHDL-A elements and mechanisms will appear in a separate glossary document.

2. Design requirements

The first numbering (DR1 to DR64) was indicated in a list of requirements prepared by Wolfgang Nebel in March 1991. A new set of requirements were gathered by Mart Altmae and numbered sequentially (DR65 to DR180). Some "holes" occur in numbering due to historical reasons. In the DOD, we retain the numbers, but we omit the origin of the requirement (North American or European) and we prefix the number with DR.

Number	Author(s)	Brief title
DR1	Mark Brown	Continuous time.
DR2	Mark Brown	Conversion between continuous and discrete time.
DR3	Mark Brown	Network of interconnected elements.
DR4	Mark Brown	Port behavior.
DR5	Mark Brown	Operators.
DR6	Mark Brown	Error tolerance.
DR7	Mark Brown	Miscellaneous requirements.
DR12	C. Le Faou et al	New key-words and electrical objects.
DR13	C. Le Faou et al	Input parameter for entities.
DR14	C. Le Faou et al	Package and user library for electrical modelling.
DR16	D. Rouquier	New key-words and underlying semantics for electrical modelling
DR17	A.S. Gilman	Analog modelling.
DR18	A.S. Gilman	Physical quantities.
DR19	A.S. Gilman	Baseline character set.
DR46	M. Altmae	Correspondence between VHDL objects and alternative representations in other netlist formats.
DR47	M. Altmae	Modify the VHDL initialization to include DC analysis for analog parts.
DR48	M. Altmae	Specified interface semantics between digital and analog parts of a design.
DR62	D. Rodriguez et al	Dimensional equations.
DR63	D. Borrionne et al	New key-words and electrical objects. Modification and review of CLF+1 (= DR12).
DR64	D. Borrionne et al	Access to the current value crossing an electrical component.
DR65	D. Rouquier et al	At least two labels.
DR66	D. Rouquier et al	Re-use digital constructs.
DR67	D. Rouquier et al	Definition of analog time.
DR68	D. Rouquier et al	Time synchronisation.
DR69	D. Rouquier	Portability of analog models.
DR70	D. Rouquier	Use of existing methods.
DR71	D. Rouquier et al	User control over digital to analog interactions.
DR72	D. Rouquier et al	User control over analog to digital interactions.
DR73	N. Whitaker	Dimensional analysis of physical types.
DR74	D. Rouquier	Domain of application.
DR75	D. Rouquier	Several modelling techniques.
DR76	T. Rahkonen	Consistency checks for excitations and results in different analog simulators.
DR77	A. Patterson	Analog modelling requirements in VHDL.
DR78	D. Rouquier	Mixed entities required.
DR79	D. Rouquier	Implementation issues.
DR176	C. Ussery	Single timing semantics.
DR177	C. Ussery	Pass-through of analog values.
DR178	C. Ussery	Fully intermixed designs.
DR179	K. Nolan	Specification of initial analog conditions.
DR180	K. Nolan	Specification of non-conserved analog systems.

3. General guidelines

This section provides general guidelines for the design of VHDL-A.

3.1. Scope of the language

DO1 VHDL-A should be suitable for the description and simulation of mixed, analog-digital, systems. No synthesis semantics is taken into account.

3.2. Digital aspects

DO2 VHDL-A will be a "super-set" of VHDL-92; any description that is valid in VHDL 92 will also be valid in VHDL-A, ***with the same simulation results***.

The only permitted exception to this is that new keywords introduced into the language may conflict with identifiers used in a VHDL 92 description.

From DR17.

3.3. Analog aspects

DO3 The analog part of VHDL-A should be targeted primarily towards the following applications:
- DC and transient analyses
- electronic circuits (OpAmps, PLLs, comparators, ...)
- lumped element systems.

From DR74.

DO4 Attention should be paid to ensure that frequency analysis is possible (for purely analog descriptions), provided that this does not have too great an impact on the language.

From DR3, DR5.

DO5 Mixed, time-frequency analysis (MFTA) could be considered.

From DR17.

DO6 The other domains (mechanical, thermal, ...) could be easily introduced as they are analogous to the electrical domain (definition of *across* and *through* quantities).

DO7 The micro-wave domain will not be taken into account.

This is contradiction with part of DR17. Contacts should be taken with MHDL.

DO8 The semantics of analog simulations will obviously be different from that of digital simulations; the syntax of analog descriptions should also be specific (to clearly identify the analog parts). Analog descriptions ought to re-use existing (VHDL 92) syntax where appropriate, for such constructs as expressions, function calls, if-then-else statements, ...

From DR66.

3.4. Mixed aspects

DO9 A few, basic, mechanisms will be provided to allow analog parts and digital parts of a mixed circuit to interact.

From DR2, DR3, DR5.

DO10 The places where such interactions occur will be explicitly described by the user.

From DR71, DR72.

DO11 The basic A/D and D/A interaction mechanisms of VHDL-A will be completely defined, and will make no use of the "foreign interface" of VHDL-92.

This not in contradiction with section 4. In other words, analog will not be "foreign" to VHDL-A.

DO12 The guidelines contained in this paragraph should ensure a reasonable degree of portability, while giving enough freedom to the implementation. The intention is to maintain portability without defining the simulation algorithm.

Further study is required on the meaning of portability. This aspect is also closely related to the time synchronisation issues (see 8.1 below).

DO13 Further study is required on the restrictions which may be required on the combination of analog and digital constructs (but note DR178).

DO14 The nature (analog OR digital OR mixed) of any part of a mixed circuit is contained in its VHDL-A description.

For example, a component configured as a digital entity-architecture pair cannot be dynamically re-configured during simulation to become analog.

4. Analog structures

DO15 Using VHDL-A, it should be possible to describe the structural composition of analog sub-circuits connected by analog wires.

This implies to determine clearly what aspects may already be found in the digital domain and what aspects are specific to the analog domain.

From DR3, DR12, DR14, DR63.

4.1. Similarities with the digital domain

DO16 The following aspects are very similar to the corresponding ones in the digital domain:
- blocks
- components
- configuration of a component onto an entity-architecture pair
- generate statements.

From DR3, DR14.

DO17 For these aspects, it should be possible to re-use the existing syntax.

From DR66.

4.2. Specific aspects of analog domain

DO18 The analog ports (provisional name : PINs), or terminals (see the glossary for the definition of a terminal), of an analog component are characterised by a voltage across the PIN and a current flowing through the PIN.

It will be necessary to specify a clear convention concerning the direction of currents (entering or leaving the components).

From DR4, DR12.

DO19 The interactions between an analog component and its environment are always "two-way" (not "in" or "out" ; the "inout" digital mode does not seem appropriate either). The term "binding" has been used.

From DR3, DR63.

DO20 The analog wires (provisional name : NODEs) of an analog structure are characterised by a voltage. The NODE object should be able to provide the values of all its contributing currents from its incidental branches (see the definition of branch in the glossary).

This assumption might not hold in micro-wave circuits, but see DO7.

From DR12.

DO21 The connection of several PINs to a NODE means that :
- all PINs must have the same voltage
- the sum of PINs currents must be zero (Kirchhoff current law, or KCL).

This assumption might not hold in micro-wave circuits, but see DO7.

From DR1, DR4, DR12, DR63, DR77.

DO22 Analog components may also need "parameters" ; parameters may be defined as generics whose value may vary during simulation.

From DR13.

DO23 Analog components could embody SPICE netlists since the analog simulator should understand this de facto standard description.

Communication between top-level (VHDL-A) description and SPICE netlist should be carefully studied (node correspondence, generics, parameters, …).

From DR46.

DO24 By changing the definition of the "across" and "through" quantities, and by keeping KCL and KVL (equal "across" quantity on a "node", null sum of "through" quantities on a node), other domains (mechanical, thermal, ...) can easily be introduced.

DO25 For these aspects, new syntax has to be created.

From DR66.

5. Analog behaviors

DO26 Using VHDL-A, it should be possible to describe the behavior of analog circuits. Two styles should be provided:
- relations or equations
- procedural models.

These two styles will form the “core" of the analog part of VHDL-A, and should be designed with great care. It might be interesting to allow mixing both styles within the same analog part, like in FAS or MAST.

From DR4, DR12, DR75, DR79.

5.1. Relations

DO27 The behavior of an analog part may be described by an un-ordered set of differential equations.

Notes:
- Each one of these equations is merged into the set of equations to be solved to perform analog simulation.
- Other equations come from the structural description.
- Adding an equation to the equation set should be complemented by adding an independent variable to the set of variables.

From DR4, DR12, DR14, DR70, DR75, DR79.

5.2. Procedural models

DO28 The behavior of an analog part may be described by a procedure, returning branch currents (*through* quantities) as a function of node voltages (*across* quantities). Each of these procedures is called, by the simulator during analog simulation, to find node voltages satisfying KCL.

It might be useful to allow a procedural model to return also voltages as functions of currents (e.g. for an inductance).

From DR4, DR70, DR75, DR79.

5.3. Access to voltages and currents

DO29 For analog behavioral modelling (and for A-to-D interactions), it is necessary to designate the voltage or the current of a PIN (seen from within a component). It is also useful to designate a branch current or a node voltage (seen from outside components).

The following notations have been proposed:

PIN.V	access to the voltage of a pin
(PIN1,PIN2).V	access to the voltage difference between PIN1 and PIN2
PIN.I	access to the current (entering or leaving) a pin
NODE.V	access to the voltage of a node
COMP1.PIN2.I	access to the current (entering or leaving) the pin of an outside component
(PIN1,PIN2).I	access to the branch current of the branch flowing from PIN1 to PIN2 (see the definition of a branch in the glossary)
(PIN1,NODE).I	access to the branch current flowing from PIN1 to NODE (see the definition of a branch in the glossary).

The above notations are provisional and purely syntactic, and do not mean that a PIN or NODE is represented by a "record" (the exact representation is left to the implementation). They may be used to allow reading, or writing, or within an equation (relation).

The notation COMP1.PIN2.I is intended to designate the current flowing into the component instance COMP1 coming from the (unique) node connected to the pin PIN2 of that instance and assuming that the corresponding model have a pin named PIN2. This current is “seen from the outside” of component COMP1 and may be read (but not written) by some other element of the enclosing architecture (port map, pin map, parameter map, or direct access by a process or an “analog black-box”).

From DR14, DR64.

6. Interactions between analog and digital domains

DO30 Interactions between the analog and digital domains address both the way the model is described and the way it is simulated.

From DR2, DR69, DR79.

DO31 The user must be given control over the exact transformations used between the analog and digital domains of description.

From DR2, DR48, DR71, DR72.

6.1. Analog to digital interactions

DO32 A user-supplied thresholding function is required to ensure that digital events are triggered only when “significant changes” occur in the analog part.

The idea behind the term “significant changes” is that at least some form of user controlled thresholding must be performed by the analog simulator in order to avoid unnecessary computing in the digital simulator. The exact sharing of work for analog to digital interactions between the analog and digital simulators is still to be defined.

One useful assumption to help produce a consistent definition of analog to digital interactions in VHDL-A would be: “The time unit used by the analog simulator when dating threshold crossings for the digital simulator must be equal to or greater than the secondary resolution time unit of the digital simulator”.

It was proposed that the analog simulator reports to the digital simulator the fact that an analog value has crossed a threshold along with the direction of the crossing (e.g. rising or falling).

From DR5, DR72.

DO33 The basic mechanisms could be restricted to :
• a digital process may read an analog value (voltage or current)
• a digital process may be made "sensitive" to an analog value (and in this case the thresholding function is used to decide when to trigger the process; this is similar to the "monitor" construct proposed by M.Brown and B.Hanna, but more general).

From DR72.

DO34 Default conversions between analog and digital domains may be provided. This would make the connection between PINs and PORTs legal without additional syntax. However, it should be illegal to connect a PIN to a SIGNAL.

From DR79.

6.2. Digital to analog interactions

DO35 A user-supplied "slope function" might be required, to avoid numerical problems due to discontinuities in the analog part. This "slope function" is similar to the "waveform generator" proposed in DR5, but less general.

The term “slope function” was considered inappropriate. The term “transition function” was proposed but does not seem to be convenient either because “transition” has a very specific semantics in VHDL. A more convenient name is still to be defined.

From DR5, DR71.

DO36 The basic mechanisms could be restricted to :
• an analog behavioral model may read a digital signal
• it would be possible to read a digital signal to drive a node by controlling either its voltage or (part of) its current, or a mix of both.

From DR71.

DO37 Default conversions between digital and analog domains may be provided. This would make the connection between PORTs and PINs legal without additional syntax. However, it should be illegal to connect a PORT to a NODE.

From DR79.

7. Units

7.1. Entities

DO38 Entities in VHDL-A may have PINs and PARAMETERs in addition to PORTS and GENERICs ; following this line, entities may be digital, analog, or mixed.

From DR78, DR177.

7.2. Architectures

DO39 Architectures in VHDL-A may contain NODEs and "Analog-Black-Boxes" (see 7.1) in addition to SIGNALs and PROCESSes ; following this line, architectures may be digital, analog, or mixed.

From DR78, DR177.

DO40 In addition to GENERIC MAP and PORT MAP, components will have "PIN MAP" and "PARAMETER MAP" ; same for blocks.

From DR63.

DO41 Analog-to-digital and digital-to-analog interactions will occur only within architectures containing PROCESSes and "Analog-Black-Boxes", and making use of one of the basic mechanisms described in 6.1 or 6.2.

From DR79.

7.3. Math package

DO42 For analog behavioral modelling a specific mathematical package is required, providing useful functions such as sine, cosine, exp, log, ln, complex numbers and associated operations.

Notes:
- Double precision (64 bits) is required.
- Piece-wise linear functions could be provided (such as arrays of value pairs).
- Matrix manipulations could be provided (such as FFT).

The design of the mathematical package may be addressed by another working group such as the VHDL Math Package Study Group.

From DR7, DR77.

DO42 Time derivative is necessary for analog behavioral modelling. It must be possible to apply it to any analog quantity:
- node voltage
- branch current
- pin voltage
- pin current
- user-defined extra independent variable (whether within a "relation" or within a "procedural model").

Derivatives with respect to other quantities than time were also considered but not retained.

From DR5, DR12, DR70, DR77.

DO43 Also an integral operator is useful (at least "from 0 to now").

7.4. Electrical package

DO44 For electrical modelling (i.e. to model analog structures), a specific electrical package is required, providing the declarations and the implementations of basic circuit elements (such as resistance, capacitance, inductance, sources, …).

Common models for semiconductors (diode, transistor) could also be provided. For transistors, we can mention SPICE level 1 to 3 and BSIM models.

From DR14.

7.5. Dimensional analysis package

DO45 The language should support dimensional analysis. To this aim, a separate package should be designed.

This objective requires to design a new syntax since the existing VHDL syntax for physical types does not seem to be convenient.

From DR18, DR62, DR73, DR77.

DO46 Only static dimensional analysis is retained, that is all dimensional verifications are to be performed at compile time only.

8. Simulation mechanisms

DO47 As in VHDL, simulation mechanisms are part of the norm, but simulation algorithms ought not to be specified. However, mixed-mode analog-digital simulation mechanisms should be defined precisely in order to ensure a reasonable degree of portability for the models written in VHDL-A.

DO48 Both electrical simulation of digital circuits (with simplified techniques like Iterated Timing Analysis or Waveform Relaxation) and precise simulation of analog circuits (with accurate SPICE-like techniques) should be possible.

From DR74.

8.1. Time management

DO49 Obviously, the analog simulation should follow a different time management than the digital simulator. However, the simulation of a mixed model requires that events on digital signals and evaluations of analog quantities be ordered along a single time axis.

From DR2, DR67, DR68, DR176.

DO50 The definition of time synchronisation between analog and digital parts should be complete and precise enough to ensure a reasonable degree of portability. In addition, it should be transparent to the user (no explicit use of “last active postponed process”).

From DR68, DR69.

8.2. Initialisation

DO51 The initialisation of a mixed model must include DC analysis for the analog parts (in addition to the normal procedure for the digital parts) ; this may involve iterations.

From DR47, DR179.

8.3. Simulation control

DO52 As an analog simulator usually implements some form of iterative methods to compute the state of the circuit, some control over the iterations is needed.

Possible controls may be:
• To specify the accuracy at which the electrical quantities are computed.
• To select specific simulation techniques or algorithms on a block-by-block basis, a block being an analog black-box for example.
• To specify the format of the analog output waveforms, possibly with some post-processing (for example, to compute the power dissipation).

Only the communication mechanism to pass these data to the simulator should be defined, not the way the simulator should handle them.

From DR16.

9. Open questions

9.1. Relations

This style of description has been illustrated in requirements DR12 and DR14, but only on simple examples (elements with only two pins, or structural equivalent circuit for the bipolar transistor). It is necessary to explore the limits of this style:
• is it possible to describe an element having three or more pins using only equations?
• what if several relations are declared within one architecture (does this describe a "structure" ; will the relations interact through nodes)?
• use of "if-then-else" within relations?
• is it necessary to explicitly declare a new variable with each new equation?

9.2. Procedural models

This style of description has been described, with very few details, in requirements DR4, DR70 and DR75. It is used in several existing simulators (FAS, MAST, also M). It is very natural if the equation-set is built using the "modified nodal method" (each node corresponds to one variable: the node voltage, and one equation : KCL). It may include such features as access to the ‘old' value of an analog quantity, access to the value of the time-step, ... It is necessary to explore also the limits of this style.

9.3. Modification to the digital simulation kernel

It will be necessary to introduce at least some modifications to the digital part of VHDL-92 to allow for well-defined mixed-mode descriptions and simulations. These modifications will emerge from the A-to-D interaction mechanisms, and from the problem of time synchronisation. The following could be sufficient:

1) The wait statement is extended to allow processes to be sensitive to analog quantities; examples:

   wait on N.V; wait on PIN1.I; wait on COMP1.PIN2.V;

   Notes:
   • the visibility rules will have to be refined
   • a thresholding function is required, associated with the analog quantity to which a process is sensitive.

   This sensitivity takes effect as explained below.

2) The simulation cycle is modified:

   2.1) When the digital "kernel" reaches the point where all processes have been run and have suspended (including "postponed" processes), instead of advancing time (to the "next digital date", which is strictly greater than the current date), the digital kernel should be capable of halting and giving control back to some "general kernel", which will take care of analog simulation.

   2.2) The digital kernel is capable of being re-started by the "general kernel" with the following order : advance time to the "next digital date" and perform normally at that date (in this case there is no real modification of the simulation cycle).

   2.3) The digital kernel is also capable of being re-started by the "general kernel" with the following order : advance time to date T0 (with T0 strictly smaller than the "next digital date") and at that time, accept "unforeseen events". These events come from the analog part, and as far as the digital kernel is concerned, they simply imply resuming the execution of one or several processes currently suspended on a "wait on N.V" statement.

   With these modifications, and with a few assumptions on the analog simulator, time synchronisation can be achieved.

9.4. Assumptions on the analog simulator

When started, the analog simulator never produces "zero-delay-events" (in other words, its time-step is never zero); this seems a reasonable assumption to place on any analog simulator, and it might be crucial when specifying time synchronisation.

The analog simulator can not tell its "next analog date" (date of the next time- step to be computed by the analog simulator). On the other hand, when computing a time-step, it can store old values (values of the previous time step); thus it can be told either to "accept" a time-step (forget old values) or to "reject" a time-step and to compute (possibly by interpolation) a shorter one. This is a soft form of backtracking.

With these assumptions, and with the above modifications on the digital simulator, time synchronisation can be achieved.

9.5. Miscellaneous questions

• Limits of the LRM: Will the LRM describe the structure of the mixed simulator, the time synchronisation mechanism, the method for building the equation-set, the method for solving it?

• Error tolerance (DR6) and portability (DR69); N.I returns 0 or epsilon?

• 64 bits? (left to the implementation?)

• Statement for "connecting" two nodes (DR177)?

• "node-set" to help DC.

Appendix A: Detailed analysis of design requirements

DR1	**Mark Brown**	**Continuous time.**
Continuous time:		Supported. Analog quantities vary continuously in time but the simulator computes them only at discrete time-points.
Analog simulation algorithms are proprietary:		Supported. The analog part of the VHDL-A LRM will be less detailed than the digital part.
Kirchhoff's laws:		Supported. Kirchhoff's laws form the basis of the semantics for analog simulation.

DR2	**Mark Brown**	**Conversion between continuous and discrete time.**
		Supported.

DR3	**Mark Brown**	**Network of interconnected elements.**
No direction:		Supported.
A node connects two or more terminals:		Supported.
VHDL 92 component statements are good for analog structures:		Supported.
N-Port theory:		Not completely understood, please provide a tutorial!
A branch between two terminals:		Branch current not unique !?
Modal and domain conversions:		??

DR4	**Mark Brown**	**Port behavior.**
KCL and KVL:		Supported.
Branch current seen as function of other analog quantities:		Supported.
Modelling by relationship:		Supported. Assuming “relationship” is the same as “equations”.

DR5	**Mark Brown**	**Operators.**
Threshold function for A-to-D:		Supported.
Threshold function for D-to-A:		Supported. To be refined.
Time derivative:		Supported.
Transformation between time and frequency, partial derivative:		??

DR6	**Mark Brown**	**Error tolerance.**
		Not understood.

DR7	**Mark Brown**	**Miscellaneous requirements.**
Math package for transcendental operations (and floating-point exponent):		Supported.

Complex numbers (rectangular or polar):	Supported. But it is second-priority for transient analysis of mixed circuits.
Physical units:	To be clarified.

DR12	**C. Le Faou et al**	**New key-words and electrical objects.**
Modeling by equation:		Supported. Adding an equation to the equation set should be complemented by adding an independent variable to the set of variables...
Key-words introducing new objects:		Supported. Essentially a re-phrasing of Kirchhoff's laws.
New type “electrical”:		Syntactic issues ("type electrical is record…”). It is perhaps too early to propose detailed syntactic forms.
Time derivative:		Supported.

DR13	**C. Le Faou et al**	**Input parameter for entities.**
		Supported. Except using parameters in the digital domain: this needs further study.

DR14	**C. Le Faou et al**	**Package and user library for electrical modelling.**
		It is too early to propose detailed syntax ; the examples are all very simple ones, and tend to suggest that this method is limited to elements having only two pins, possibly parameters, and no internal states. To be studied further.

DR16	**D. Rouquier**	**New key-words and underlying semantics for electrical modelling.**
		Not considered (too long, too syntactical ; the main ideas have been rephrased in subsequent requirements by the same author).

DR17	**A.S. Gilman**	**Analog modelling.**
		Not completely understood.
Micro-wave circuits:		Not supported.
Use of “concurrent” (global?) variables to model analog parts:		Not supported.
Analog part operates in parallel with the digital one:		Supported.
Simulation algorithm is implementation-defined:		Supported.
Upward compatibility:		Supported.
‘DOT for time derivative:		Interesting alternative to d-dt()…

DR18	**A.S. Gilman**	**Physical quantities.**
		Not completely understood.
Dimensional analysis in the language:		Supported.

Floating-point values for physical quantities:	Supported.

DR19	**A.S. Gilman**	**Baseline character set.**
		Not considered (already considered by VHDL-92).

DR46	**M. Altmae**	**Correspondence between VHDL objects and alternative representations in other netlist formats.**
		Not completely understood.
A-to-D and D-to-A interactions:		Seems to suggest A-to-D and D-to-A interactions by direct connection of (VHDL) signals to (SPICE) nodes ; this seems the only way during the interim phase (using the "foreign interface" before VHDL-A is standardised) but it will not be the case in VHDL-A.
Programs to adapt SPICE libraries to VHDL-A:		Supported.

DR47	**M. Altmae**	**Modify the VHDL initialization to include DC analysis for analog parts.**
		Supported. But we should find a way to avoid iterations…

DR62	**D. Rodriguez et al**	**Dimensional equations.**
Dimensional analysis in the language:		Supported.
Detailed syntax:		Too early.

DR63	**D. Borrionne et al**	**New key-words and electrical objects. Modification and review of CLF+1 (= DR12).**
Connecting several PINs to a NODE does not imply the same semantics as connecting several PORTs to a SIGNAL:		Supported.
Proposed syntax:		Too early.

DR64	**D. Borrionne et al**	**Access to the current value crossing an electrical component.**
Need to access branch currents:		Supported.
Proposed syntax:		Too early.

DR65	**D. Rouquier et al**	**At least two labels.**
		Not considered (not technical; in fact, as there will be two different standards, it will be easy to check whether a given simulator is conformant to 1076 or to 1076.1).
Static criterion about circuit type:		The criterion to tell whether a circuit is digital, analog, or mixed will be known only after elaboration.

DR66	**D. Rouquier et al**	**Re-use digital constructs.**
		Supported. In fact, to be monitored during the detailed design of the language.

DR67 D. Rouquier et al **Definition of analog time.**

Supported.

DR68 D. Rouquier **Time synchronisation.**

Conversion between analog and digital time:	Supported.
Unique time axis:	Supported.
Proposed strategy:	To be studied in greater details ; such a model (of the mixed simulator) must be viewed as conceptual, otherwise we might enter domain of "over-specification".

DR69 D. Rouquier **Portability of analog models.**

To be analysed by vendors of mixed simulators!

DR70 D. Rouquier **Use of existing methods.**

General:	Supported.
Modified Nodal Method:	Over-specification?
Procedural models:	Supported. To be refined.
Extra equations (and associated extra variables):	Supported.
Old value, time-step:	Supported. See procedural models.
Time derivative:	Supported.

DR71 D. Rouquier et al **User control over digital to analog interactions.**

General:	Supported.
Details:	Depend on the exact definition of analog behavioral modelling in VHDL-A. The problem of discontinuity may be solved by the "slope function" associated with any signal used (in read-only mode) in the analog part.

DR72 D. Rouquier et al **User control over analog to digital interactions.**

Supported.

DR73 N. Whitaker **Dimensional analysis of physical types.**

Supported.

DR74 D. Rouquier **Domain of application.**

Supported.

DR75 D. Rouquier **Several modelling techniques.**

Supported. The syntax of "analog procedural models" may be similar to that of digital processes, but the semantics will be completely different.

DR76 T. Rahkonen **Consistency checks for excitations and results in different analog simulators.**

Not understood. This text seems to refer to a situation where different simulators are used in a very loosely coupled way, and the simulation of the whole circuit requires a lot a manual (and error-prone) operations, to feed the results of one simulator as stimuli to another simulator. It does not seem to envisage true mixed-mode simulation in VHDL-A.

DR77 A. Patterson **Analog modelling requirements in VHDL.**

Supported. Perhaps phase-angle descriptions are second priority.

DR78 D. Rouquier **Mixed entities required.**

Supported.

DR79 D. Rouquier **Implementation issues.**

Supported.

DR176 C. Ussery **Single timing semantics.**

Extensions to the timing model (due to analog):	Should be incorporated within VHDL 92.
Achieve a single simulation cycle mechanism:	Supported.

DR177 C. Ussery **Pass-through of analog values.**

Supported.
Assuming that the mixed component in the associated example will have a mixed interface (at least two PINs, possibly some ports) and will contain, in addition to its ("purely digital") behavior, a "connect statement" (still to be invented) to indicate that the two nodes (coming from the source and going to the reader) are connected.

DR178 C. Ussery **Fully intermixed designs.**

Supported.
"runtime statements" : to be clarified); closely linked to DR177.

DR179 K. Nolan **Specification of initial analog conditions.**

Supported. Also user-supplied "hints" before DC are usefull (to help DC converge faster).

DR180 K. Nolan **Specification of non-conserved analog systems.**

Not understood. Please provide tutorial!

Appendix B: Summary of submitted requirements

The purpose of this document is to summarize all requirements submitted to the sub-par 1076.1 for analog extentions to VHDL. It includes the ones relating to analog matters, which were already submitted to the VHDL'92 restandardisation effort, as presented by Kevin Nolan in his document on consolidated North American and European Analog VHDL Requirements, the additional ones collected by Mart Altmae, and finally a few new North American requirements, presented by Kevin Nolan at the VHDL-Forum meeting in Santander last April.

1. Support for Continuous Time

Ref reqs: A1, E48

Summary: The execution of a VHDL model must be extended from the notion of the evaluation of events at discrete points in time to the notion of continuous evaluation over continuous time.

2. Mixed mode support: conversion between analog and digital domains

Ref reqs:

Summary: Objects defined using discrete time semantics (signals) and continuous time semantics (waveforms) must be able to com- municate with one another to provide the ability to perform mixed-mode simulation.

3. Support for analog pins and a way to connect them

Ref reqs: A2, E48

Summary: Electrical circuits are typically described as an interconnection of elements that exhibit some electrical behavior. The connection points at the elements are defined in terms of ports. Each electrical port consists of two physical connection points that are defined as terminals (N-Port Theory).

4. Analog pin behavior conforming to conservation laws

Ref reqs: A3, E13, E16

Summary: The objective is to describe the physical operation of circuits. The most common method is the use of Kirchoffs current and voltage laws. We therefore speak of electrical behavior in terms of currents flowing through the branches and the voltages across nodes of a network

5. New operators and intrinsics to support domain conversion and the specification of derivatives

Ref reqs: A5

Summary: To support the conversion from analog domain to digital domain a threshold function is needed in order to schedule events on the digital side. Conversely, a analog waveform generator function is needed to support the conversion of digital signals to analog waveforms. In order to model certain analog devices, a time-derivative operator is required.

6. Specification of error tolerances to be used by numerical algorithms used to solve the analog system

Ref reqs: A6, E16

Summary: Since analog simulation employs numerical methods in order to solve a system of ordinary differential equations, an error tolerance must be specified to bound the numerical errors generated by these methods.

7. New data types and associated operators

Ref reqs: A7

Summary: Several new data types are required to model various systems. Complex numbers and/or phasors (polar coordinates) are needed to model filter networks, etc. Similarly, transcendental functions are required to model semiconductors.

8. Specification of generics whose value can be dependent on time or other system variables

Ref reqs: E13, E16

Summary: It is necessary for modeling non-linear devices (e.g. semiconductors) to define models having the value of certain parame- ters vary as a function of time or the state of other simulation variables.

It is necessary, for modelling non linear components (diode, transistor, ...), to define models having some inputs to be considered as parameters whose value is varying during simulation, according to the state of the electrical circuit. (A well-known example is, in the Ebers and Moll bipolar transistor model, to define a current source whose value depends on the value of an other current) The introduction of input parameters is a very limited syntactic modification, because the syntax of generics can be kept for parameters, the key-word "generic" being replaced by "parameter".

The syntactic analysis should be the same for parameters as for generics. The elaboration phase should be different, but it is easier to have a different keyword according to different objects (generic and parameter) in state of having two categories of generics whose differenciation during the elaboration phase would be tricky.

As an example, an entity "linear resistor" could be described as:

```
entity RESISTOR  generic (VAL:REAL);
   port (P1, P2:pinelec ELECTRICAL); -- see E12
about pinelec
end RESISTOR;
```

An other entity non linear resistor could be described as:

```
entity NL_RESISTOR  parameter (VAL:REAL);
   port (P1, P2:pinelec ELECTRICAL);
end RESISTOR;
```

and the interconnection of such components could be described either with generic map or with parameter map. (see requirement E14)

Remarks:

This new kind of object is also necessary for non electric models (for example delay modelling at logic level when delays can vary during simulation).

This parameter being only an input parameter, cannot generate global variables.

9. Definition of new pin types and semantics to be specified by users

Ref reqs: E16, E24, E48

Summary: To model systems containing non-electrical or hybrid devices, it must be possible to define custom pin types. For example, rotational pins would be used to model motors, thermal pins to describe self-heating electrical systems, etc. The conserved properties of these custom pin types most also be specified.

10. Ability to reference the through, (e.g. current) and across (e.g. voltage) components of analog entities

Ref reqs: A5, E16

Summary: In conserved systems, behavior is described by specifying the relationship between dependent variables of the system to the independent variables. In electrical systems, onemight specify how the current through a device is a related to the voltage across its terminals. This requires access to the current through terminals and the voltage across them.

11. Ability to specify the conservation laws associated with a user defined pin type

Ref reqs: E16

Summary: For every type of pin that can be connected to other pins in a network, the properties of that pin which must be conserved has to be supplied. In electrical systems, the usual conservation laws are given by KCL, and KCL. Analogous conservation laws must be specified for non-electrical systems.

12. Specification of error tolerances for a specific node

Ref reqs: E16

Summary: It is desireable to be able to specify the maximum relative error when using KCL to calculate the current through a specific node. According to KCL the current through any node will always sum to zero. The maximum relative error at that node would be the ratio of the difference between the currents flowing into the node to the currents flowing out.

13. Specification of and event trigger threshold to cause evaluation of a event driven process body

Ref reqs: E16

Summary: In order to determine whether an digital event has occurred as a result of an analog state change, it is desirable to allow the specification of the minimum analog change threshold to con- trol the evaluation of a process body. A higher value for this parameter would cause fewer evaluations of the process body and therefore faster simulation, a lower value would result in more evaluations, longer simulations, and possibly greater simulation accuracy at the analog transition points.

14. Allow specification of a foreign simulator interface

Ref reqs: E24

Summary: To allow VHDL to communicate with foreign (non-VHDL) simulators, it is required to allow the specification of a foreign simulator interface.

15. Inclusion of DC operating point analysis as part of time-zero initialization and a way to conditionaly execute blocks based upon the initialization state.

Ref reqs: E47

Summary: Electrical systems have what is called a dc operating point. This describes the quiescent state of the system at time-zero. Usually, before the simulation of a transient behavior, the dc operating point must be computed. To accommodate this, the initialization semantics of VHDL must be extended to include dc analysis. It is also necessary to control the execution of behavior blocks based upon the initialization state. In other words, it must be possible to only execute certain blocks at the dc operating point and at no other time.

16. Analog Modeling

Ref reqs: A17

Summary: To allow the description of analog system/circuit behavior from the following perspectives: discrete-event models of digital circuits, continous-time descriptions of analog circuits, frequency-domain descriptions of analog circuits and wave-domain descriptions of microwave circuits.

17. Physical quantities

Ref reqs: A18, E62

Summary: Support for dimensional analysis of physical quantities.

18. Introduction of simultaneous equations

Ref reqs:

Summary: relation this key-word introduces relation between currents and voltages for any edge of the actual electrical network. It is an equation (not an assignement and this cannot apply to a signal).

19. Access to current value crossing an electrical component

Ref reqs: E64

20. At least two labels

Ref reqs: E65

Concerning whether or not any given simulator is conformant with IEEE-1076,it is desirable to have at least two different labels :

- one for purely digital circuits : "VHDL-1076-92"
- one for mixed, analog-digital, circuits : "AHDL-1076.1-93"
- possibly one for purely analog circuits, depending on how 1076.1 is specified.

Also it is desirable to have a clear (static) criterion to identify a circuit (configuration, or entity plus architecture) as being digital, analog, or mixed.

21. Re-use digital constructs

Ref reqs:

Summary: Whenever an analog (or mixed) model uses the same kind of construct as a digital one, it is desirable to use exactly the same syntax and semantics as already defined in IEEE-1076-92, to avoid misunderstandings.

Examples of constructs which can be shared by digital and analog :

- type definition, function definition, sequential procedure definition,
- function or procedure call, ...

Examples of constructs which are different in syntax and in semantics :

- connection of several ports to a signal versus connection of several pins to a node ;
- relation (or analog procedural model) versus digital process.

22. Definition of analog time

Ref reqs: E66

Obviously, to perform mixed (analog-digital) descriptions and simulations, clear definitions are needed for the digital time (this already exists) and for the analog time (to be defined). Moreover the definition of the analog time should not be completely opposed to that of the digital time, to allow conversion and synchronisation.

Digital time is defined in a very precise way by the LRM. In any given simulation, digital time may be viewed as the list of dates (each one expressed as an integer number of femto-seconds) at which at least one event occurred. Between two successive dates, nothing happens and signals keep their last value.

Analog time could be defined in a similar way : it is the list of dates at which the simulator decided to compute new values for (some of) the voltages and currents. Between two successive dates, currents and voltages are, strictly speaking, unknown, but can be evaluated by interpolation. Whether each of the dates in the list is expressed by a real number (in seconds) or by an integer number (in femto- seconds) is not crucial : in fact most existing simulators could input and output dates using a VHDL-like format (integer number of femto-seconds) even if real numbers are used internally.

Following this line, the opposition between "discrete" time and "continuous" time disappears.

23. Time synchronisation

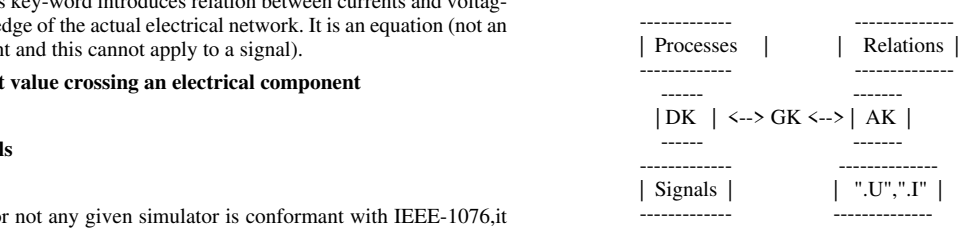
Ref reqs: E67

Obviously, to perform mixed (analog-digital) descriptions and simulations, clear definitions are needed for the conversion between analog and digital times. This problem is often cited and sometimes viewed as a simple problem of "type conversion" : converting a digital date (an integer number of Femto-seconds) to and from an analog date (viewed as some kind of real number, possibly in seconds). See for example "Functionnal modelling of analog blocks in VHDL ; Bernt Arbegard ; Euro-Vhdl-91, Stockholm".

This conversion is necessary but not sufficient ; a definition of how digital dates and analog dates will be intermixed along the (unique) time axis is also needed. This is the problem of time synchronisation, which requires that at least a broad model of a complete mixed simulator be described : digital kernel (conformant with 1076-92, plus "hooks"), analog kernel (to be defined) and interactions between analog and digital (also to be defined).

Assume a definition of analog time such as the one proposed in a companion requirement (provisional number rqr-ana-4) ; use a template of the mixed simulator (see next page) ; define a general strategy for running and suspending the analog and digital parts ; this is linked to the definition of "hooks" in the digital part (see next page) ; address the problem of fine synchronisation (see for example "Mixed VHDL - Analog Simulation, Mart Altmae, Euro-Vhdl-91, Stockholm").

Proposed sketch of the mixed simulator :



On the left-hand side is the digital part, compliant with VHDL-92. It is composed of processes, signals, and the digital kernel (or DK for short). It knows the list of "analogue relevant signals" : those signals which are used, in one way or another, in the analogue part (see companion requirement rqr-ana-8).

On the right-hand side is the electrical part. It is composed of differential equations, nodes and pins, and an electrical kernel (or AK for short). It knows the list of "digital relevant voltages (or currents)" : those voltages (or currents) which are used, in one way or another, in the digital part (see companion requirement rqr-ana-9).

The general kernel (GK) is in charge of synchronisation between the analog part and the digital part.

It is desirable to introduce as few new features as possible in the DK, and to place as few assumptions as possible on the AK.

Step 1 : the DK performs normally, for some date t1, as specified in IEE1076-87; when it reaches the step where "time may advance", instead of advancing time (to t2), it suspends and reports to the GK with the following information : list of (new) values of analogue relevant signals, date of next digital event (here t2) ;

Step 2 : the GK forwards this information to the AK, and starts it ; the AK should not go beyond t2 ; it may suspend before t2 (let us say at time t3) if some of the digital relevant voltage has crossed one of its relevant threshold ; the AK then suspends and reports to the GK with the following information : list of (new) values of digital relevent voltages, list of those voltages which have crossed a threshold, present date (t2 or possibly t3).

step 3 : the GK forwards this information to the DK and starts it ; we can go back to step 1.

New features in the DK : when it reaches the situation where "time may adavance", the DK must be capable of being suspended and restarted, possibly at some earlier date with "unforeseen" events triggered by the analogue part.

Assumptions on the AK : it must be capable of calling user-supplied thresholding functions on the digital relevant voltages and of adjusting its time-step to halt before t2.

Dissymmetry of the interaction : this sketch is not symmetrical : the DK may be re-started "earlier than foreseen" (at t3) while the AK is only instructed "not to go beyond t2" ; the reason is that digital signals do not change between events, while analogue voltages must be viewed as continuous.

NOTE : what to do at time 0 is also to be defined. See for example requirement E47.

24. Portability of analog models

Ref reqs: E68

Summary: In the digital domain, the semantics is defined in such a way that models are portable : the same simulation leads to the same events happening at the same dates, whichever simulator is used.

Portability is also desirable for analog (and mixed) models, but must be defined in a different way, to account for the fact that solving a set of differential equations is not an exact science!

- Define a "precision" and state that two analog simulations are equivalent if the values of any voltage or current at any date (whether actually computed or interpolated) are the same (within that precision).
- Normalize the way the equation-set is built : Starting from the structural and functionnal description of the different parts of the circuit, a single equation set will be built (at compilation or elaboration time) and solved (at simulation time) ; the contents of this equation set must be normalised.
- Specify the computation accuracy used for solving the equation set
- Normalize the interactions between the digital and analog parts
- Leave the rest to the implementation (detailed method used for solving the equation set, time-step management, ...).
- (Define a "validation suite" ?)

25. Use of existing methods

Ref reqs: E70

The structural constructs (pins and nodes, or others) and the functionnal constructs (relations, or others) used in analog models, and the implied semantics (KCL, equations, others, ...), must be such that the building and solving of the set of differential equations may be performed using existing, proven methods.

Review existing methods, as implemented in existing simulators ; choose a minimum set of features to be specified and normalised.

Possible items:

- modified nodal method for building the equation set
- procedural models providing the branch current as a function of voltages ;
- implicit states (one more equation) when the analytical solution is unknown
- access to "old" value of a variable, and to the corresponding time-step
- use of time derivative
- others ...

26. User control over Digital to Analog interactions

Ref reqs: E71

Summary: Interactions from the digital part to the analog part should be user-definable; this will allow technology-specific or application-specific interactions to be modelled at the desired level of abstraction; it will not prevent from using "standard" packages for standard cases.

As soon as the principles of analog functionnal modeling are defined ("relations", or procedural models) it will be easy to specify how the (up-dated) values of signals may be used inside analog models.

Possible interactions :

- the value of an enumerated-type (digital) signal may be used within an "if" or "case" statement to choose between several forms of a differential equation ;
- the value of a real or integer signal may be used as a parameter within a differential equation ;
- both may be used in the (sequential) computations taking place within a procedural model (if such a modeling technique is used).

The problem of discontinuity must be addressed (i.e. a controlled voltage source which suddenly swtiches from 0V to 5V may disturb the analogue simulator).

27. User control over Analog to Digital interactions

Ref reqs: E72

Summary: Interactions from the analog part to the digital part should be user-definable ; this will allow technology-specific or application-specific interactions to be modelled at the desired level of abstraction ; it will not prevent from using "standard" packages for standard cases.

The values of any node voltage or any pin current should be usable in the right-hand side of signal (or variable) assignement statements in the digital part ; they should also be usable in boolean expressions.

The digital processes (or equivalent processes) which make use of such current or voltage values will have to be triggered when the digital conditions are met (e.g. event on a signal in the sensitivity list), and in addition they will have to be triggered also when a "significant" change occurs on one of the current or voltage they use.

To avoid a great number of unnecessary calls to these processes, some form of thresholding, with user-provided functions, should be included in the analog kernel.

The problem of "time-step adjustment" must be addressed (see for example "Mixed VHDL-Analog Simulation, Mart Altmae, Euro-Vhdl-91, Stockholm").

28. Dimensional analysis of physical types

Ref reqs: E73

Summary: Extend the VHDL type model to support the dimensional analysis of physical quantities

It should be possible to write expressions involving physical quantities without having to overload the standard arithmetic operators. For example, it should be possible to multiply a variable of ‘type’ voltage by one of ‘type’ current and assign the result to something of ‘type’ power without having to define an overloaded multiplication operator.

This requirement re-states two requirements from the 1076 requirements ballot. Specifically E62 by D. Rodriguez and D. Rouqier and A18 by Alfred S. Gilman.

All physical quantities or units of measurement have dimensions defined in terms of seven basic physical units (metre, kilogramme, second, ampere, kelvin, candela and mole). These units are termed the ‘SI’ units and are defined by the standard ISO-1000.

For example, the unit of power, the watt, can be defined:

$2\ -3\ 1$ W = 1 kg m s (Kilogramme, metre squared, per second cubed)

If, when any physical type is defined, the dimensions of the type (in terms of the 7 basic SI units, which are perhaps defined using some standard package) are also specified then any assignment operations in the language can be checked for dimensional equivalence.

The dimension specification could be added to the declaration of a physical type, using some suitable syntactic convention for expressing the various numerators and demoninators.

Some rules for arithmetic operations (such as those in 1076 requirement E62) would be required to define how dimensioned quantities are arithmetically combined.

29. Domain of application

Ref reqs: E74

Summary: The mixed language to be specified in IEEE1076.1 should enable the user to perform not only electrical simulation of digital circuits, but also analog simulation of "truely" analog circuits.

In the first case, the electrical (and timing) performances of a digital circuit (e.g. adder, register) may be checked, and simplified simulation techniques may be used.

In the second case, the behavior of an analog circuit (e.g. operational amplifier) may be checked, and precise simulation my be necessary.

Clearly both styles must be possible when using IEEE1076.1

Check that no language feature prevents from modeling truely analog circuits.

Possible problems :

- definition of (analog) time (variable time-step must be possible) ;
- definition of "DC" analysis to be performed at time 0.

30. Several modeling techniques

Ref reqs: E75

Summary: The mixed language to be specified in IEEE1076.1 should provide the user several modeling techniques for the analog parts.

Modeling by equations has already been proposed (see requirement E12). This method is simple and, in some cases, necessary. But it is well known that it leads to a greater number of equations in the equation-set.

Procedural description should also be provided. In this method, analog blocks are seen as controlling their branch currents as functions of the node voltages (the node voltages are iteratively proposed by the analog kernel when solving the equation set, until the branch currents returned by analog blocks statisfy KCL).

The "equation" method must be maintained and specified, because it is the only applicable method in some cases, but the "procedural" method must be added, because it will enable faster simulations.

Study and specify the "analog procedural model" :

- semantics (sequence of computation, with node voltages being read as input parameters, and branch current being returned as results) ;
- instants at which the analog procedural model will be called by the analog "kernel" ;
- syntax (could re-use most of the syntax of the digital process) ;

31. Consistency check for excitations and results in different analogue simulators

Ref reqs: E76

Summary: Signal typing and comparison possibility would be needed to be sure that input and output signals of different type of analogue simulators are consistent.

In analog design, variety of simulation tools must usually be used (high-level programming languages for system behavioural verification, linear ac simulators for frequency responses and time domain simulators for non-linear effects) and the main problem seems not be the actual simulation but being sure that what you have simulated is what you have in the schematics. Simulating small sub-blocks, either linear or non-linear, is possible by describing appropriate input signals by hand to that particular simulator, but verifying the correctness of the whole circuit cannot usually be done: you cannot be sure that connections between sub-blocks (especially between analog and digital parts) have the right polarity and so on.

It seems obvious, that VHDL itself is useful only in very simplified behavioural time domain analysis of analog circuits. What I would consider very useful is that different architecture descriptions of VHDL could be used as links or source code to different analog simulators that may run parallel with digital VHDL simulation. VHDL itself would be used as a design framework which takes care that the types and waveforms of input and output signals remain consistent, no matter what simulator is used.

Firstly, I'm sorry to confess that I have no practical experience of using VHDL, so these are mere guesses which may not be practical at all.

The strong signal typing capability of VHDL could be used to distinguish which signals are clocks, which ac-inputs or outputs and which mere dc voltage or current references. VHDL would check that current reference type signal is described by dc type source IBIAS in Spice, for example.

Due to different accuracies of simulators, perhaps a visual comparison would be best. The framework would show e.g. transient results of a behavioural model to be used in VHDL simulation and from a Spice simulation on same time scale, and the designer's job is to verify that they agree close enough. Thus VHDL could be used to write spesifications of the function and to test that specs are fullfilled.

32. Analogue Modelling Requirements in VHDL

Ref reqs: 77

The following items would be required to give adequate analogue support in VHDL.

- Differential Equation specification for physical units eg di/dt, dv/dt
- New standard physical type definitions to cover current and power units
- A standard set of functions for often-used analogue sources eg sine, cosine, voltage/current controlled sources
- New VHDL signal type definitions that can contain current, voltage and phase angle information.
- Overloading of standard operators to support real and imaginary numbers (ie to handle phase-angle summations)

By inventing some functions and new types many analogue issues can be addressed in the current 1076 standard. The most important issue is that their should be agreement on the names and definitions so that everyone adopts them.

33. Mixed entities required

Ref reqs: 78

Summary: When using 1076.1, it should be possible to describe and simulate mixed (analog plus digital) circuits, circuits having digital I/Os (e.g. bits or bit-vectors, whether input, output or three-state) and analog I/Os (e.g. analog input or analog output).

Thus it should be possible to declare an entity as having, at the same time, (digital) PORTS and (analog) PINS (and possibly GENERICS and PARAMETERS).

34. Implementation issues

Ref reqs: 79

Summary: When defining 1076.1, attention should be paid to what is (easily ?) implemented and what is impossible (or nearly impossible) to implement. This should be done with the cooperation of people having an experience in the implementation of mixed simulators.

Example of feasible things :

- model an analog black box by a set of equations
- model an analog black box by a procedural model
- read a digital signal and use it in the analog behavior (with visibilty rules)
- read an analog voltage and use it in the digital behavior (with visibilty rules)

Examples of things which should be forbidden (by the syntax)

- overwrite a digital signal from within an analog black box
- overwrite an analog voltage from within a process
- turn a voltage-source into a current-source and vice-versa

35. Single Timing Semantic

Ref reqs: A176

Summary: Description: Any extensions to the semantic timing model of VHDL should be incorporated into the existing timing model of the 92 standard or should handle the model used in the 92 standard. In particular, the evaluation of continuous-time models should be integrated with the evaluation of event-driven models into a single "simulation cycle" mechanism.

36. Pass Through of Analog Values

Ref reqs: A177

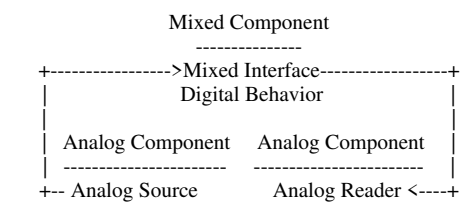
Summary: The transferal of analog values through a communication channel which passes through a mixed component which contains digital elements but does not involve mixed digital/analog operations on those values should not require a conversion to the digital domain.

37. Fully Intermixed Designs

Ref reqs: A178

Summary: Any extended language should allow the intermixing of any runtime statements, structural elements, or declarative statements within any of the structural blocks of the language. There should be no distinction between analog blocks/components/entities and digital blocks/components/entities.

Consider the following case:



The analog reader should see the exact value generated by the analog source.

38. Specification of Initial Analog Conditions

Ref reqs: A179

Summary: The initialization of a VHDL model must be extended from the current procedural notation to something that allows the inclusion of initial analog characteristics.

A method for specifying the initial (time-zero) conditions of analog models must be provided. Examples of possible initial conditions are the charge on a capacitor, the flux of an inductor. More generally, an initial condition can be some voltage across or some current though a branch. If an initial condition is not specified, the simulator will use the value calculated from the DC solution of the circuit as the value for time-zero at a branch when transient analysis is performed.

39. Specification of Non-conserved analog systems

Ref reqs: a180

Summary: It should be possible to specify non-conserved analog systems in VHDL. This would extend the applicability of VHDL to incude control system design.

It should be possible to specify a non-conserved analog system to VHDL. A non-conserved (e.g. control) system use signal flow semantics to interconnect functional blocks. In other words, the signals form a directed graph of functional blocks. A functional block implements some function on the input signal(s), relating them to the output signal(s).

40. Facilitate Top Down Design

Ref reqs: A181

Summary: VDHL must facilitate a top down design methodology for analog systems from very high levels of abstraction (e.g. H(s), Control Loops, etc.) down through their physical implementations. The must include the specification of S domain expressions and related transfer functions.

41. Support for Sample Data Systems

Ref reqs: A182

Summary: VHDL must support the simulation of Sampled Data Systems and other DSP applications. The must include the specification of Z domain expressions and related transfer functions.

42. Modeling of Analog Noise

Summary: VHDL must be able to model analog noise sources and allow them to be conditionally injected into the behavior of a model.

43. Modeling of an ideal delay

Ref reqs: A183

Summary: VHDL must be able to model an ideal analog delay.

44. Modeling of ideal voltage and current sources

Summary: VHDL must be able to model ideal voltage and current sources.

Specificly, VHDL should support (at least) the following kinds of devices:

- voltage-controlled-voltage source
- voltage-controlled-current-source
- current-controlled-current-source
- current-controlled-voltage-source

45. Speciification of next time step

Ref reqs: A184

Summary: VHDL must allow the specification, from within a model, of the next time step the simulator is guaranteed to evaluate.

46. Additional Operators and Intrinsics

Ref reqs: A185

Summary: VHDL must support the following functions and operators:

- an intergal operator which allows the specificating of an initial condition
- power-of operator (** in FORTRAN)
- random number function

47. Support for Frequency Domain Modeling

Ref reqs: A186

Summary: VHDL-A should support frequency domain modeling and analysis. Similarly, it should support small-signal AC analysis.

- transfer function whose arguments are the coefficients of S or Z poly's

Requirement Number: E12

Current Status:
Chapter Priority:
ISAC Cross Reference:
Requirement Cross Reference: E11 E13

Functional Area(s): Analog
Functional Area Priority: Essential
Submitted By: Claude Le Faou and Jean Mermet
claude@athena.imag.fr
Date Submitted: 12/7/90
History:
Brief Title:New key-words and electrical objects.
Summary:
Description of Requirement:
VHDL 87 Solution:
Suggested Strategy:

-1- New key-words introducing a Block of simultaneous equations relation this key-word introduces relation between currents and voltages for any edge of the actual electrical network. It is an equation (not an assignement and this cannot apply to a signal).

-2- Key-words introducing new objects

parameter See requirement E13

pinelec non directional electrical formal interface port. A characteristic of this kind of object is to support two informations:a current (which has to be oriented) and a voltage (with respect to a reference voltage). These informations are real values functions of a real value time. The type of a pinelec object has to be electrical. node allows connection between electrical ports and represents equipotential. The type of a node is electrical. The semantic linked to a node is to force the voltage of the node to be equal to the voltage of all the pinelec connected to the node,and to force the sum of all the currents of the pinelec connected to the node to be equal

to zero.

-3- New type

electrical record type associated with node and pinelec objects.

type electrical is record V,I:real:end record;

-4- Derivation operator

d_dt operator defining the derivative with respect to time.

Requirement Number: E13

Current Status:
Chapter Priority:
ISAC Cross Reference:
Requirement Cross Reference: E11 E12

Functional Area(s): Analog
Functional Area Priority: Essential
Submitted By: Claude Le Faou and Jean Mermet
claude@athena.imag.fr
Date Submitted: 12/7/90
History:
Brief Title:Input parameter for Entity
Summary:
Description of Requirement:
VHDL 87 Solution:
Suggested Strategy:

It is necessary, for modelling non linear components (diode, transistor, ...), to define models having some inputs to be considered as parameters whose value is varying during simulation, according to the state of the electrical circuit. (A well-known example is, in the Ebers and Moll bipolar transistor model, to define a current source whose value depends on the value of an other current) The introduction of input parameters is a very limited syntactic modification, because the syntax of generics can be kept for parameters, the key-word "generic" being replaced by "parameter".

The syntactic analysis should be the same for parameters as for generics. The elaboration phase should be different, but it is easier to have a different keyword according to different objects (generic and parameter) in state of having two categories of generics whose differenciation during the elaboration phase would be tricky.

As an example, an entity "linear resistor" could be described as:

```
entity RESISTOR  generic (VAL:REAL);
    port (P1, P2:pinelec ELECTRICAL); -- see E12
about pinelec
end RESISTOR;
```

An other entity non linear resistor could be described as;

```
entity NL_RESISTOR  parameter (VAL:REAL);
    port (P1, P2:pinelec ELECTRICAL);
end RESISTOR;
```

and the interconnection of such components could be described either with generic map or with parameter map. (see requirement E14)

Remarks:

This new kind of object is also necessary for non electric models (for example delay modelling at logic level when delays can vary during simulation).

This parameter being only an input parameter, cannot generate global variables.

Requirement Number: E14

Current Status:
Chapter Priority:
ISAC Cross Reference:
Requirement Cross Reference:

Functional Area(s): Analog
Functional Area Priority: Essential
Submitted By: Claude Le Faou and Jean Mermet
claude@athena.imag.fr
Date Submitted: 12/7/90
History:
Brief Title:Package and User Library for Electrical modelling
Summary:
Description of Requirement:Building of electrical_lib:library of primitive electrical components
VHDL 87 Solution:None
Suggested Strategy:

```
package electrical_modelling is
    type ELECTRICAL is record V,I:REAL;
end record;

function F_DIODE (IS,VJ:REAL) return REAL;

component RESISTOR  generic (VAL:REAL);
    port (P1,P2: pinelec ELECTRICAL);
end component;

component CAPACITOR  generic (VAL:REAL);
    port (P1,P2: pinelec ELECTRICAL);
end component;

component INDUCTOR  generic (VAL:REAL);
    port (P1,P2: pinelec ELECTRICAL);
end component;

component VOLTAGE_SOURCE  generic (VAL:REAL):
    port (P1,P2: pinelec ELECTRICAL);
end component;

component CURRENT_SOURCE  generic (VAL:REAL);
    port (P1,P2: pinelec ELECTRICAL);
end component;
```

Remarks:

- two categories of primitive components are described:linear components with generic values
- (above), and non linear components with parameter values (below)

```
component NL_RESISTOR  parameter (VAL:REAL);
    port (P1,P2: pinelec ELECTRICAL);
end component;

component NL_CAPACITOR parameter (VAL:REAL);
    port (P1,P2: pinelec ELECTRICAL);
end component;

component NL_INDUCTOR  parameter (VAL:REAL);
    port (P1,P2: pinelec ELECTRICAL);
end component;

component NL_VOLTAGE_SOURCE  parameter (VAL:REAL);
    port (P1,P2: pinelec ELECTRICAL);
end component;

component NL_CURRENT_SOURCE  parameter (VAL:REAL);
    port (P1,P2: pinelec ELECTRICAL);
end component;
end electrical_modelling;
package body electrical_function is
function F_DIODE (IS,VJ: REAL) is
begin
    return  IS*exp(40*VJ - 1);
end;
end electrical_function;
```

```
-- library of primitive electrical components
    use electrical_modelling.all;
    entity RESISTOR
        generic (VAL: REAL);  -- linear resistor
        port (P1,P2: pinelec ELECTRICAL);
    end RESISTOR;

    architecture Behavior of RESISTOR is
    begin
        relation
            P1.V - P2.V - VAL * P1.I = 0;
            P1.I + P2.I = 0;
        end relation;
    end Behavior;

    entity NL_RESISTOR  parameter (VAL:REAL); -- non-
    linear resistor
        port (P1,P2: pinelec ELECTRICAL);
    end NL_RESISTOR;

    architecture Behavior of NL_RESISTOR is
    begin
        relation
            P1.V - P2.V - VAL * P1.I = 0;
            P1.I + P2.I = 0;
        end relation;
    end Behavior;

    entity CAPACITOR generic (VAL: REAL);  -- linear
    capacitor
        port (P1,P2: pinelec ELECTRICAL);
    end CAPACITOR;

    architecture Behavior of CAPACITOR is
    begin
        relation
            VAL * (d_dt (P1.V - P2.V)) -  P1.I = 0;
            P1.I + P2.I = 0;
        end relation;
    end Behavior;

    entity NL_CAPACITOR parameter (VAL: REAL); -- non-
    linear capacitor
        port (P1,P2: pinelec ELECTRICAL);
    end NL_CAPACITOR;
    architecture Behavior of NL_CAPACITOR is
    begin
        relation
            VAL * (d_dt (P1.V - P2.V)) -  P1.I = 0;
            P1.I + P2.I = 0;
        end relation;
    end Behavior;
    entity INDUCTOR  generic (VAL: REAL);   -- linear
    inductor
    port (P1,P2:pinelec ELECTRICAL);
    end INDUCTOR;

    architecture Behavior of INDUCTOR is
    begin
        relation
            P1.V - P2.V - VAL * (d_dt (P1.I)) = 0;
            P1.I + P2.I = 0;
        end relation;
    end Behavior;

    entity NL_INDUCTOR  parameter (VAL: REAL); -- non-
    linear inductor
        port (P1,P2: pinelec ELECTRICAL);
    end NL_INDUCTOR;

    architecture Behavior of NL_INDUCTOR is
    begin
        relation
            P1.V - P2.V - VAL * (d_dt (P1.I)) = 0;
            P1.I + P2.I = 0;
        end relation;
    end Behavior;

    entity VOLTAGE_SOURCE  generic (VAL:REAL);-- linear
    voltage source
        port (P1,P2:pinelec ELECTRICAL);
    end VOLTAGE_SOURCE;

    architecture Behavior of VOLTAGE_SOURCE is
    begin
        relation
            P1.V - P2.V - VAL = 0;
            P1.I + P2.I = 0;
        end relation;
    end Behavior;

    entity NL_VOLTAGE_SOURCE  parameter (VAL: REAL);--
    nonlinear voltage source
        port (P1,P2: pinelec ELECTRICAL);
    end NL_VOLTAGE_SOURCE;

    architecture Behavior of NL_VOLTAGE_SOURCE is
    begin
        relation
            P1.V - P2.V - VAL = 0;
            P1.I + P2.I = 0;
```

```
    end relation;
end Behavior;

entity CURRENT_SOURCE  generic (VAL: REAL);-- lin-
ear current source
    port (P1, P2:pinelec ELECTRICAL);
end CURRENT_SOURCE;
architecture Behavior of CURRENT_SOURCE is
begin
    relation
        VAL - P1.I = 0;
        P1.I + P2.I = 0;
    end relation;
end Behavior;
entity NL_CURRENT_SOURCE  parameter (VAL:REAL);--
nonlinear current source
    port (P1,P2:pinelec ELECTRICAL);
end NL_CURRENT_SOURCE;
architecture Behavior of NL_CURRENT_SOURCE is
begin
    relation
        VAL - P1.I = 0;
        P1.I + P2.I = 0;
    end relation;
end Behavior;

Example of user description

library electrical_lib;
use electrical_modelling.all;
entity SIMPLE_TRANSISTOR is
    generic (  VR1, VR2, IE0, IC0, ALFAN, ALFAI,
               VC1,VC2: real);
    port ( EM, BAS, COL: pinelec ELECTRICAL);
end SIMPLE_TRANSISTOR;

architecture behavior of SIMPLE_TRANSISTOR is
node BI, CI:ELECTRICAL;
function  VJ (I01, V1, ALFA, I02, V2:REAL) is
begin
    return (F_DIODE (I01, V1) + ALFA * F_DIODE
(I02, V2));
end;
begin
    R1:RESISTOR  generic map (VR1)-- linear resis-
tor
        port map(BAS, BI);
    R2:RESISTOR generic map (VR2)-- linear resistor
        port map(CI, COL);
    C1:CAPACITOR generic map (VC1)   -- linear ca-
pacitor
        port map(BI, EM);
    C2: CAPACITOR generic map (VC2)   -- linear ca-
pacitor
port map(BI,CI);
    J1:NL_CURRENT_SOURCE   -- non linear  current
source
        parameter map  (VJ(IE0, (BI.V-EM.V), ALFAN,
IC0, (CI.V-BI.V))
        port map(BI, EM);
    J2:NL_CURRENT_SOURCE   -- non linear  current
source
        parameter map  (VJ(IC0, (CI.V - BI.V), AL-
FAI, IE0, (BI.V - EM.V))
        port map(BI, CI);
end behavior;
```

Requirement Number: E16

Current Status:
Chapter Priority:
ISAC cross reference:
Requirement Cross Reference:

Functional Area(s): electrical modeling, "real" signals.
Functional Area Priority:
Submitted By: D. Rouquier , CNET , BP98 38243 Meylan:
tel:+ 33 76 76 41 31:fax:+ 33 76 90 34 43
Date Submitted: 12-Feb-91
History:

Brief Title:new key-words (and underlying semantics) for electrical modelling.

Summary:nodes and pins for electrical structure, relations for electrical behaviour, physical types for R, C, V, ...

Description of requirement:propose electrical counter-parts to signals, ports and processes

VHDL 87 solution:OK for physical types, not really feasible for Kirchoff's

law.

Suggested strategy:

-1- Introduction

This requirement proposes a number of new constructs for electrical modeling. The main ideas have been first proposed by J. Mermet, D. Borrione, C. Le Faou and D. Rodriguez, and this text contains mainly refinements to their propositions. Please refer to requirements E12, E13, E14 by J.Mermet and C. Le Faou, and to requirements "access to the current value crossing an electrical component" and "new key-words and electrical objects" by D. Borrione, C. Le Faou and D. Rodriguez. Here "electrical" simply means that Kirchoff's law must be verified

throughout the circuit:no provision is made to introduce a difference between DC and transient analysis, or to provide small-signal analysis. Some ideas have been taken from an experiment:"simulation of analog circuits using VHDL" to be presented by D. Rodriguez in the VHDL Forum

(Marseille, April 91).

-2- Structural descriptions

-2-1- Nodes and pins

For the structural description of interconnected electrical (or analog) components, it is desirable to retain the current style of VHDL87. Because ports and signals have a very precise semantics, well adapted to digital, but no so good for electrical , it is proposed to create two new concepts, called node and pin, and which would be as follows:

A pin is the electrical counter-part of a port in the digital world. A pin is a means for analog components to interact with other analog parts. A pin has a current aspect and a voltage aspect. Declaring a pin is similar to declaring a port, e.g.:

```
entity ex1 is
    pin ( A, B: electrical );
end ex1;
```

except that a different semantics is implied. A pin has no "direction" (in, out, inout) because the interaction happening through it is always bidirectional.

A node is the electrical counter-part of digital signals. A node is used to interconnect two or more pins. Declaring a node is similar to declaring a signal, e.g.:

```
architecture A of some_entity is
    node A, B:electrical;
begin ...
```

except that a different semantics is implied.

The specific semantics implied is:when interconnecting several pins on the same node, the V parts of all pins must be equal and the I parts must sum to zero (equal

and zero must be defined, e.g. within 1E-6, see -4-1-). The new key-word electrical which looks like a type is only intended to allow further extensions to mechanical or thermal domains, but the idea is that, unlike the types used for signals and ports, the type used for nodes and pins should not be user- definable. In fact it should be defined only by the associated semantics and by the imeter and vmeter

functions (see below). The reasons are the very specific semantics (which would be meaningless if complex, user-definable types were allowed on nodes and pins) and the freedom which must be given to the implementation.

With these new concepts, assembling analog components is as easy as

assembling digital ones:

```
entity EX1 is end;
architecture A of EX1 is
    node AA, BB, CC:electrical;
component R
generic ( rvalue:resistance:= 1 kohm );
pin ( A, B:electrical );
end component;
```

```
component C
generic ( cvalue:capacitance:= 1 pF );
pin ( A, B:electrical );
end component;
begin -- A
R1:R generic map (2 kohm);pin map ( AA , BB ):
C1:C pin map ( BB, CC );
...
end A;
```

Two functions are pre-defined:imeter and vmeter, to allow reading the current and voltage aspects of pins and nodes:whether the nodes and pins have a "record" structure with a voltage field and a current field is left for implementation. imeter (some-node) returns zero (or something smaller than some epsilon, e.g. 1e-6) imeter (some-pin) returns the current through that pin vmeter (some-pin) returns the

voltage of that pin vmeter (some-node) returns the voltage of that node, which must be the same as that of any pin connected to the node. The convention for imeter will have to be stated (entering the pin or going out of the pin). Designating a pin might look like "componentname.pinname" , e.g.:

```
architecture A of some_entity is
    signal x:current;signal y,z:voltage;
    node AA, BB:electrical;
component R is
    pin ( A , B:electrical);
end component;
begin
    ... ...
    R1:R pin map (AA , BB );
    x <= imeter(R1.A);
    z <= vmeter(BB);
    y <= vmeter(R1.B);-- same as z
    ...
end A;
```

These functions can be used within structural descriptions (to feed a current- controlled voltage source with the current on some pin) or within behavioural descriptions (-3-1-). Their value may be assigned to a (digital) signal of the proper type (-3-2-).

-2-2- Mixed (analog+digital) models

It is desirable to allow the description of mixed (analog+digital) models. With the nodes and pins this is easy, for it is possible to declare pins and ports on the same entity, or signals and nodes in the same architecture. This will certainly be useful for conversion components to be inserted between the digital an analog worlds:it may also be a simple way of defining variable components (e.g. variable resistor or variable voltage source). Examples;

```
entity V_R is -- variable resistor, controlled by
R_Value
    port ( R_Value in resistance );
    pin ( A, B:electrical );
end V_R;
```

here a port (with its present-day syntax and semantics) is used to control the value of the resistance, and two pins are used to represent the electrical behaviour:the architecture (-3-1-) should reflect the fact that " Va-Vb = R_Value * I ". Not e; this "port" has no physical counter-part in the circuit ...

```
entity digital_to_analog_interface is
    port ( d:in bit );
    pin (a, b:electrical );
end;
```

the idea here is to model an interface block to be used as a transformer between a digital signal and an electrical node. The digital signal is fed through a port of type bit, and the electrical node is controlled through a couple of pins;the architecture could drive these pins with a voltage source, or with a current source, or ...

```
entity analog_to_digital_interface is
    port ( s:out bit );
    pin ( e:electrical );
end analog_to_digital_interface;
```

this example is symmetric of the previous one:a pin "e" is sensed (possibly without sinking any current) and a port "s" is driven (for example in accordance with vmeter(e) ... ...). Note;in the two above examples, the ports do have physical counterparts.

-3- Functional

descriptions After having described a circuit as black-boxes into black-boxes, there comes a time when one must describe the function, or behaviour, of some basic black-boxes. For the digital world the "process" is a very powerful construct;-something similar should be defined for electrical behaviour.

-3-1- Relations

As proposed by C. Le Faou, a relation is used to specify a set of equations which must be fulfilled inside an electrical component (Kirchoff's law, which must be fulfilled between components, is indicated by the key-words pin and node). It is assumed that the electrical VHDL simulator will gather all these equations to make up the set of equations to be solved during simulation. A relation is not oriented, this allows a greater modeling freedom, and the use of different simulator strategy (either v=f(i) or i=f(v)).

example 1 , "normal" resistor:

```
entity fixed_R is
    pin (a, b:electrical );
end;
architecture A of fixed_R is
begin
    relation
        imeter(A)+imeter(B) = 0 A;
        vmeter(A) - vmeter(B) - r_value * imeter(A)
= 0 V;
    end relation;
end A;
```

remark:"= 0" might be implicit, each line in a relation would describe a quantity which must be zero (with some precision)

-3-2-Mixed models

It is natural to use, within the same architecture, signals and processes for the digital part, together with nodes and relations for the electrical part. Precise rules will have to be stated concerning the interaction between digital and electrical. This interaction occurs when a relation uses the value of a signal, or when a process uses the imeter or vmeter functions. The problem is to clearly specify when the process will be triggered, and when the set of equation will be up-dated.

Example 1, going from digital to electrical;

```
entity digital_to_analog_interface is
    port ( d:in bit );
    pin (a, b:electrical );
end;
architecture a of digital_to_analog_interface is
begin
    relation
        imeter ( a ) + imeter (b ) = 0.0;
        case d is
            when '1' => vmeter(a)-vmeter(b)= 5 v;
            when '0' => vmeter(a)-vmeter(b)= 0 v;
        end case;
    end relation;
end a;
```

here a bit "d" is used to switch between two (sets of) equations:when "d" changes, will the electrical part be up-dated immediately or at the next time-step ? Remark:-this type of interface would introduce a discontinuity, which some simulators may want to avoid. Applying a continuous voltage (e.g. a slope) would require sequential features in relations (to be clarified).

Example 2:going from electrical to digital:

```
entity analog_to_digital_interface is
    port ( s:out bit );
    pin ( e:electrical );
end analog_to_digital_interface;
architecture a of analog_to_digital_interface is
begin
    relation
        imeter(e)=0.0;-- high impedance probe...
end relation;
    s <=    '1' when vmeter(e) > 3 v else
            '0' when vmeter(e) < 2 v else s:
 ------------------------- Schmidt trigger ..
end a;
```

here the current is set to zero to model a "high-impedance" probe, and the voltage on "e" is used to control the "s" (digital) signal. One question to be asked is "when will the equivalent process driving s be triggered ?" at each electrical time-step (for chances are that the voltage on e will change, even by a very small amount) or each time a "significant" change occurs on vmeter(e) ? Paragraph -4-2- proposes something.

Example 3 , variable component;

```
entity current_controlled_voltage_source is
    generic ( k:resistance );
    port ( I:in current );
    pin ( a , b:electrical );
end;
architecture a of current_controlled_voltage_-
source is
begin
    relation
        imeter ( a ) + imeter (b ) = 0.0;
        vmeter(a)-vmeter(b)= i*k;
    end relation;
end a;
```

here the relation used for describing the voltage source is a very simple one:nevertheless it shows the use of a generic (fixed during simulation) and of a port (may vary during simulation).

The following example shows how to actually control this current_controlled_-voltage_source.

```
entity EX1 is end;
architecture A of ex1 is
component ANY is
    pin ( P1, P2, P3:electrical);
end component;
component current_controlled_voltage_source is
    generic ( k:resistance );
    port ( I:in current );
    pin ( a , b:electrical );
end component;
    node n1, n2, n3, n4, n5:electrical;
    signal s1:current;
    begin
        A1:any pin map ( n1, n2, n3 );
        C1:current_controlled_voltage_source
               generic map ( 1 kohm );
               port map ( s1 );
               pin map (n4, n5 );
        s1 <= imeter ( A1.P2 );
        ...
end A;
```

here the (digital) signal S1, with its present-day definition, is well adapted to describe the one-way interaction from A1 to C1.

Remark:this two-stage technique (the current on A1.P1 drives the digital signal S1, and then S1 is used inside C1) may imply extra delta delays:for this reason the "parameter" and parameter map " constructs proposed by C. Le Faou could be better.

-4- General considerations

-4-1- Electrical precision

It is desirable to give the user control over the computation accuracy for the electrical simulation:this is a feature supported by some electrical simulators. It might be specified by:

node a, b, c:electrical, precision (current:1]A , voltage:1mV)

: whether this is absolute precision, or an upper bound, is still to be defined.

-4-2- Events on real signals

This is not specific to electrical modeling. In VHDL87, assuming that process P1 drives signal S1, of type "real", and that process P2 is sensitive to S1 (and not to S1'transaction), it would be safer to specify that P2 must be triggered only each time a "significant" change occurs on S1. This could be attached to S1 itself, and described by:

signal S1:real precision (1E-6);

this would be of interest even for purely "digital" models (no

Kirchoff's law) and is different from the precision proposed in -4-1-.

-4-3- Physical types

In order to ease electrical modelling and to make models more consistent, it is proposed to "hard-wire" inside VHDL the equivalent of a package defining resistance, current, capacitance, ... and the related operations U=RI, Q=CV, ... This is feasible in VHDL87, except that physical types should use 64 bit precision ....

Example:

```
type resistance is range 0 to 1e9
units mohm;-- milli-ohm
    ohm = 1000 mohm;
    kohm = 1000 ohm;
    megohm = 1000 kohm;
end units;
...
function "*" ( r:in resistance:i:in current ) re-
turn voltage;
function "*" (  i:in current;r:in resistance) re-
turn voltage;
function "*" ( c:in capacitance;v:in voltage ) re-
turn charge;
function "*" ( r:in resistance;c:in capacitance )
return time;
...
```

One problem is to deal with the innumerable new units introduced by time derivation ...

-4-4- Concurrent relations ?

It is proposed to allow several relations within the same architecture, and to consider this as structural composition (the same is true today for several processes in the same architecture). This requires further study.

The following example shows two resistors in series, with two

```
architectures which should be equivalent.
    entity two_res is
    pin ( a , b:electrical );
end;
architecture a of two_res is
    constant r1:resistance:= 1 kohm;
    constant r2:resistance:= 2 kohm;
begin
    relation
        imeter(A)+imeter(B) = 0.0;
        vmeter(A) - vmeter(B) =(r1+r2) * imeter(A);
    end relation;
end a;
architecture b of two_res is
    constant r1:resistance:= 1 kohm;
    constant r2:resistance:= 2 kohm;
    node x:electrical precision ... ....
begin
    r1:relation
        imeter(A)+imeter(x) = 0.0;
        vmeter(A) - vmeter(x) =r1 * imeter(A);
    end relation;
    r2:relation
        imeter(x)+imeter(b) = 0.0;
        vmeter(x) - vmeter(b) =r2 * imeter(x);
    end relation;
end b;
```

it is a bit far-fetched, because it is meaningful only if one makes the following assumption;reference to imeter(x) inside relation R1 (or R2) does not yield zero:instead it yields the current through an implicit pin R1X implicitly connected to node X ...

Parallel resistors are below:

```
architecture c of two_res is -- parallel
    constant r1:resistance:= 1 kohm;
    constant r2:resistance:= 2 kohm;
begin
    relation
        imeter(A)+imeter(B) = 0.0;
        vmeter(A) - vmeter(B) =(1/(1/r1+1/r2)) *
imeter(A);
    end relation;
end c;
architecture d of two_res is
    constant r1:resistance:= 1 kohm;
    constant r2:resistance:= 2 kohm;
begin
    r1:relation
        imeter(A)+imeter(B) = 0.0;
        vmeter(A) - vmeter(B) =r1 * imeter(A);
    end relation;
    r2:relation
        imeter(A)+imeter(B) = 0.0;
        vmeter(A) - vmeter(B) =r2 * imeter(A);
    end relation;
end d;
```

here an implicit addition of currents should be assumed:this addition is consistent with the pin and node semantics exposed in -2-1-, provided the "implicit pin" assumption is also applied when relation R1 (or R2) refers to imeter(a) or imete r(b):"a", which is a pin of entity two_res, can be viewed as a node inside architecture A:this is quite in line with the way processes see ports ... ...

On the other hand, the two references to imeter(a) (one in R1, one in R2) should not return the same result, which is not so consistent. This requires further study.

Requirement Number: E24

Current Status:
Chapter Priority:
ISAC Cross Reference:
Requirement Cross Reference:

Functional Area Priority:
Functional Area(s): General/Analog
Submitted By: Andreas Hohl, ah@ZTIVAX.SIEMENS.DE
Date Submitted: 91/03/14
History:
Brief Title:Mixed-mode design with VHDL
Summary:

Within a hardware system design it is required to handle analog parts, beside the digital parts of the system, as well. VHDL is designed for digital systems, neither for analog ones. What will be the general strategy to handle mixed-mode (digital/ analog) designs with VHDL.

Description of Requirement:

VHDL should handle the design of hardware systems, where a system may contain various parts (digital, analog, software, optical, mechanical, thermal and others). If it is not possible to describe one of these aspects with VHDL, interfaces should be determined between VHDL and dedicated languages, which are able to handle it. As the author is aware, this requirement is hard to fulfil in total. But for the VHDL users as well as the tool suppliers it is of great importance to know

in advance the direction the standard HDL VHDL is aimed to go. If digital hardware system design is the application field of VHDL now, what will it be in the future.

In a first step it should be decided how to deal with mixed-mode design. This solution should be conform with a general strategy. From the authors' point of view it is very important to define a general strategy for "VHDL for System Design" now.

VHDL 87 Solution:None

Suggested Strategies:

1) Extension of the Language

VHDL can be added by additional language features to describe analog aspects of a system to be of importance.

2) Definition of Language Interfaces

Interfaces of VHDL to an analog language should be defined.

Pros and Cons:

1) Extension of the Language

Pros:- 1 Language which handle all aspects

Cons:- Language becomes too complex

- Impact on Tools (Loss of performance, Portability/Interoperability)

2) Definition of Language Interfaces

Pros:- "Easier" maintainability/use of VHDL

- Stepwise addressing of the system design aspects

Cons:- Multiple Languages

Note:A mixed strategy should be chosen to conquer the task of system design.

Constraint:

VHDL as the standard HDL claims to be a system design language. The

users have to live with the standard (at least for additional 5 years,

until 1997).

Requirement Number: E47

Current Status:
Chapter Priority:
ISAC Cross Reference:
Requirement Cross Reference:

Functional Area(s): Mixed digital/analog
Functional Area Priority:
Submitted By: Mart Altmae
Date Submitted: 25/1/91
History:
Brief Title:Modify the VHDL initialisation to include DC analysis for analog parts.
Summary:
Description of Requirement:

Analog simulators require a DC analysis to be performed before a timedomain solution may be obtained. This has to be added to the prescribed initialisation procedure. Note that in general an iteration is required, since in general analog signals are dependant on initial values of VHDL ports and vice versa.

VHDL 87 Solution:

Suggested Strategies:

Requirement Number: E48

Current Status:
Chapter Priority:
ISAC Cross Reference:
Requirement Cross Reference:

Functional Area(s): Mixed digital/analog
Functional Area Priority:
Submitted By: Mart Altmae
Date Submitted: 25/1/91
History:
Brief Title:Specified interface semantics between digital and analog parts of a design.
Description of Requirement:

The semantics of the communication between a digital description description, described using some logic type in discrete time, and an analog description using real valued signals (voltage, current, charge etc...) in continuous time has to be specified. The specification must be general enough to allow any user defined logic type together with any user specified type conversion function between the logic and the analog signal values. In the direction digital to analog this seems fairly straightforward, while in the other direction both conversion from continuous to discrete time, and mechanisms to detect when predefined levels are passed, have to be addressed.

Brief title:Domain of application
Summary:Description of requirement:

The mixed language to be specified in IEEE1076.1 should enable the user to perform not only electrical simulation of digital circuits, but also analog simulation of "truely" analog circuits.

In the first case, the electrical (and timing) performances of a digital circuit (e.g. adder, register) may be checked, and simplified simulation techniques may be used.

In the second case, the behavior of an analog circuit (e.g. operational amplifier) may be checked, and precise simulation my be necessary.

Clearly both styles must be possible when using IEEE1076.1

VHDL 92 solution:Suggested strategy:

Check that no language feature prevents from modeling truely analog circuits.

Possible problems:

- definition of (analog) time (variable time-step must be possible):

- definition of "DC" analysis to be performed at time 0.

Consistent with:Different from:Opposed to:

==========

fax: + 33 76 90 34 43
email:rouquier@cns.cnet.fr Date submitted:October-91 History :
Brief title:Several modeling techniques
Summary:Description of requirement:

The mixed language to be specified in IEEE1076.1 should provide the user several modeling techniques for the analog parts.

Modeling by equations has already been proposed (see requirement E12). This method is simple and, in some cases, necessary. But it is well known that it leads to a greater number of equations in the equation-set.

Procedural description should also be provided. In this method, analog blocks are seen as controlling their branch currents as functions of the node voltages (the node voltages are iteratively proposed by the analog kernel when solving the equation set, until the branch currents returned by analog blocks statisfy KCL).

The "equation" method must be maintained and specified, because it is the only applicable method in some cases, but the "procedural" method must be added, because it will enable faster simulations.

VHDL 92 solution:Suggested strategy:

Study and specify the "analog procedural model":

- semantics (sequence of computation, with node voltages being read as input

parameters, and branch current being returned as results):

- instants at which the analog procedural model will be called by the analog

"kernel":

- syntax (could re-use most of the syntax of the digital process):

Consistent with:Different from:Opposed to: