VHDL-A: JADE vs OPAL proposals
(IEEE 1076.1 Standard)

Dr. Yannick Hervé
Dipl.-Ing. Bruno Böttcher
Laboratoire ERM/PHASE
herve@erm1.u-strasbg.fr

Contents

1 Introduction **3**

2 The proposals : Description **3**
2.1 Rationale . . . 3
2.2 New types . . . 3
2.3 New definition of time . . . 4
2.4 Analog objects . . . 5
2.5 Analog behavior . . . 7
2.6 Analog connexions . . . 10
2.7 Mixed netlist . . . 11
2.8 Mixed simulation . . . 12
2.9 Analog discontinuities . . . 13
2.10 Interpretation Domain (ID) . . . 15

3 Syntax and semantic analysis **15**
3.1 Rationale . . . 15
3.2 New type . . . 15
3.3 Time definition . . . 15
3.4 Analog objects . . . 16
3.5 Analog behavior . . . 16
3.6 Analog connexions . . . 20
3.7 Mixed netlist . . . 20
3.8 Mixed simulation and Analog discontinuities . . . 21
3.9 Interpretation Domain . . . 22

4 Completeness of presentation **22**
4.1 Quality of documentation . . . 22
4.2 Adherence to VHDL-A Design Objective Document (DOD) and Design Guidelines . . . 22
4.3 General conformance to VHDL'93 . . . 25
4.4 Status of completion . . . 25

5 The user point of view **25**
5.1 Code length . . . 25
5.2 Understanding the code . . . 25
5.3 Modeling multi level descriptions . . . 26

6 Flexibility - future needs **26**

7 General recommendations **26**

8 Conclusion **27**

A Annex **29**

B Used documents **29**

B.1 OPAL 29

B.2 JADE 30

1 Introduction

This paper compares the OPAL and JADE proposals for the future IEEE 1076.1 standard from a syntactical and semantical point of view.
After a purely descriptive part, we give the pros and cons of each proposition. We analyze its conformance to the available requirement documents. We finish with some remarks and suggestions issued through the comparative study of the propositions.
This document does not replace an in extenso reading of the original documents. We focus on the divergent points of the two propositions and do not mention the points that cause no problems. Neither the underlying mathematical problems due to the problem of solving a system of differential equations nor the theory about the implementation of effective algorithms for the analog solver will be treated.

2 The proposals : Description

2.1 Rationale

The huge modification of a standard like VHDL involves the redefinition of a certain number of general principles.

JADE Propositions for modifications in the LRM are enclosed in 18 chapters of the VHDL RATIONALE. We will illustrate some of them.
In the VHDL RATIONALE, JADE intends to class all concurrent statements in three categories, structural, digital and analog concurrent statement families. This classification allows to associate the different parts to the two simulation kernels in a more transparent manner. Furthermore the sequential instructions are classified in three families called neutral, digital and analog. The subprograms can also be neutral, digital or analog. Each family has associated a set of rules which allow to simplify the descriptions in the LRM.
The attributes are classified in four categories: neutral, digital, mixed and analog. These categories reflect the responsibility of each kernel concerning the updates of the objects. These notions are important for simplifying the presentation of the interactions of the two simulation modes.

OPAL The RATIONALE is fragmented into the documentation of the proposition. The proposed modifications are strongly associated to the presented points.

2.2 New types

The JADE proposal defines some new types.

	JADE proposal	**OPAL proposal**
COMPLEX	replaces 1076-1993 floating point, composed of a real part and an imaginary part. Object of complex type has 'real_part and 'imaginary_part attributes	
REAL	subtype of Complex with imaginary part=0.0 constraint	nothing changed vs. VHDL
ANALOG	subtype of Complex. An object of analog type is dimorphic: Time or DC point=> analog with real constraint, frequency analysis => no constraint (true complex)	

Table 1: New Types

2.3 New definition of time

The VHDL 93 language does not clearly state about the time, especially it is nowhere explained when a VHDL event occurs. Integer multiples of the resolution limit are supposed. Therefore no clear basis for further definitions is laid.
Both proposals follow the White Papers about time: The TYPE time is defined in package STANDARD. It is a VHDL 93 physical type, that is, a collection of physical units is associated to the type. The current simulation time in VHDL 93 is of type "physical time", the resolution limit is "1 fs". The representation of this type is integer based and thus offers a constant absolute precision over the entire range.
Analog solvers use generally a double floating point representation of time, with constant relative precision. Since usually relative time, centered about the actual ASP (Analog Solving Point), is used, very high precision can be achieved.
The predefined floating-point type Universal_real is used as a template for the time type.
JADE uses directly the Universal_real type for TIME, OPAL declares a new type Universal_time, which are the simulation times for the analog solver.
The base unit of Universal_time is the second.
The functions presented in table 2 must be defined in package STANDARD.

JADE The time is available through a NOW impure function call.

OPAL The time is available through a NOW impure function call for "digital" time and the quantity ANOW for "analog" time.

JADE proposal	OPAL proposal
JADE use of universal_real as type for time	OPAL defines new universal_time type of type universal_real
Needed conversion functions: universal_to_physical_time(Universal_real) returns time	universal_to_physical_time(Universal_Time) returns Time
N.A.	universal_to_real_time(Universal_Time) returns Real
physical_to_universal_time(Time) returns Universal_real	physical_to_universal_time(Time) returns Universal_Time
N.A.	real_to_universal_time(real) returns universal_time
VHDL STD.standard.NOW function is recast on universal_to_physical_time($T_{c(urrent)}$)	VHDL STD.standard.NOW function is recast on universal_to_physical_time($T_{c(urrent)}$)
ANOW (Analog NOW) is defined as an impure function. In original layout no impure functions were allowed in equations. Therefore in JADE impure functions **are** allowed in equations.	ANOW declared as quantity constrained to universal_to_real_time($T_c(urrent)$).
The overloading functionality of VHDL is used and ANOW is defined as NOW, the compiler must determine the correct substitution.	
	Required precision: +-1E308, 14 digits ∀ floating point types (IEEE Std. 754 double precision)

JADE proposal	OPAL proposal
	To avoid the need for big time intervals OPAL recommends the use of T modulus (1/omega) for cyclic processes depending of (omega * T), thus avoiding big values for T.

Table 2: Definition of time

2.4 Analog objects

The two propositions define objects that allow the modelisation and simulation of analog behavior. The analog "signals" are transported through them.

The basic analog object is the QUANTITY, that acts like an analog variable. OPAL calls such an object a **free quantity**.
The user has access to the differential (with respect to time) through an implicitly declared quantity called Q'DOT in OPAL and Q'DDT in JADE. Likewise there is an access to the integrated quantity through another implicitly declared quantity Q'INTEG (for t>=0). Those quantities are read only. Array structures or records of quantities are available.
To each quantity an implicitly declared quantity Q'DELAYED(T) can be bound, which corresponds to a copy of Q delayed about the time T. The time T is approximated as closely as possible to the granularity of the analog simulator.
To model and simulate conservative systems (ruled by Kirchhoffs law) it is necessary to associate a quantity that behaves like a current (through quantity) and a quantity that behaves like a voltage (across quantity). Are defined as a NATURE the associated types bound to the keywords ACROSS and THROUGH. The definition of a nature N implicitly declares a reference node N'REFERENCE such that N'REFERENCE'ACROSS =0.0. The attributes 'ACROSS and 'THROUGH applied to the nature itself return the type of the parameter of the same name.

JADE NATURE is an extension of the type mechanism. Associated types in NATURE declaration are ANALOG types. A QUANTITY can be declared in a PACKAGE and is of subtype ANALOG. A NODE is defined as the connexion point of several branches. An explicitly declared node is called TOP NODE. It can be declared everywhere a signal can be. The declaration of a node N implicitly declares the N'ACROSS and N'THROUGH quantities. The declaration of a BRANCH defines a connexion between two NODES. If a branch B is declared between the nodes N1 and N2 of nature N, two quantities B'ACROSS and B'THROUGH are available and their types are correlated to those of the nature N. The QUANTITIES of the BRANCH are the unknowns in the behavioral equations. If the declaration of the BRANCH is made with only one node, then it is a BRANCH between this node and N'REFERENCE. [N1,N2] defines an implicit branch between the nodes N1 and N2. The two associated QUANTITIES are accessible with [N1,N2]'ACROSS and [N1,N2]'THROUGH. The last calculated value (last ASP) for the quantity Q by the analog solver is accessible through the attribute Q'INERTIA. Q'INERTIA = Q'DELAYED(T) with T equal to the time-difference between NOW and the last computational step. All declarations can have a supplied default expression. The otherwise used default expression is O'ZERO. The attribute 'ZERO returns the adequate zero value, for the used type.

OPAL NATURE is a new mechanism. The composite declaration of composite NATURE (array, records) is also allowed. Associated types in NATURE declaration are REAL types
The QUANTITIES can not be declared in PACKAGES and are of subtype REAL.
A TERMINAL is declared as the connexion point of different branches. It can be declared everywhere a signal can be. The declaration of a terminal implicitly creates a quantity (THROUGH) T'CONTRIBUTION that represents the algebraic sum of all the THROUGH quantities arriving on T.

A branch QUANTITY qualifies one or several quantities (THROUGH and ACROSS) connected to two terminals. The types of the quantities are defined by the nature of the terminal. The branch QUANTITIES are the unknowns in the behavioral equations.
All declarations (not TERMINAL) can be extended by a default expression. The default initialization value is 0.0.

	JADE proposal	**OPAL proposal**
NATURE	TYPE N IS NATURE ACROSS analog; THROUGH analog; end NATURE N;	NATURE N IS real ACROSS real THROUGH;
N'ACROSS	returns the type of the ACROSS aspect	returns the type of the ACROSS aspect
N'THROUGH	returns the type of the THROUGH aspect	returns the type of the THROUGH aspect
N'REFERENCE	is a node[1] N0 of NATURE N with N0'ACROSS=0.0	is a node[2] N0 of NATURE N with the ACROSS aspect stuck to 0.0

Table 3: Nature declaration and attributes

	JADE proposal		**OPAL proposal**
QUANTITY	QUANTITY name: analog_type := 2.0;	QUANTITY (free)	QUANTITY name: REAL := 2.0;
BRANCH	BRANCH B: subtype_nature IS N1 TO N2 := 2.0; ACROSS quantity is affected by default expression. [N1,N2] denotes an implicitly declared branch	QUANTITY (branch)	QUANTITY QA1 ACROSS T1 TO T3; QUANTITY QT1 THROUGH T1 TO T3; QUANTITY I1,I2 THROUGH V1 ACROSS T1 TO T2; each aspect can be initialized separately.
B'ACROSS B'THROUGH	ACROSS and THROUGH quantities of the branch	N.A.	N.A.

Table 4: Quantity

[1] see table 5
[2] see table 5

	JADE proposal	OPAL proposal
	NODE N1 : N := 3.0;	**TERMINAL** T1: N; can not be initialized
N1'ACROSS	The ACROSS QUANTITY of the node	N.A.
N1'THROUGH	The THROUGH QUANTITY of the node	N.A.
T1'REFERENCE	N.A.	defines a branch across quantity between T1 and reference terminal (defined by nature N'REFERENCE).
T1'CONTRIBUTION	(see Analog Behavior)	THROUGH quantity of (= sum of all contributions incident to) terminal T1

Table 5: Node/Terminal declaration and attributes

	JADE proposal		OPAL proposal
'DDT	Q'DDT generates a quantity that contains the differentiated quantity with respect to time	'DOT	Q'DOT generates a quantity holding the differentiated quantity with respect to time
'INTEG	generates a quantity holding the integrated quantity with respect to time	'INTEG	generates a quantity that contains the integrated quantity with respect to time
'INERTIA	returns value of last ASP[3]	N.A.	
'DELAYED (T)	the quantity is shifted (as near as possible) about T in time	'DELAYED (T)	the quantity is shifted (as near as possible) about T in time
'ZERO	value with which each uninitialized object is affected. Returns the zero of appropriated type (integer, floating point, scalar or composite)	N.A.	

Table 6: Other implicit quantities

2.5 Analog behavior

The propositions allow to express analog behavior through differential and algebraic equations. These equations are composed of operators applied on explicit or implicit quantities. The behavior can be described in concurrent or sequential description domain.
The base mechanism for modeling analog behavior is the equation as a simultaneous statement. The used quantities determines if it is a differential or algebraic equation. Furthermore it is possible to describe a simultaneous statement by a procedural form.
The objective of the simulator is to find the state where all equations are true. When all equations are satisfied, a precedently defined precision achieved, then a so called Analog Simulation Point (ASP) is found. The order in which the concurrent statements are written does not influence the results of the simulation. It is possible to create conditionally these instructions through a GENERATE statement.

JADE

Concurrent domain The analog equation has three forms, built around the == operator. The three possible forms, simple, conditional and selective are shown in table 7. Each equation has a TARGET which

[3] Analog Simulation Point

defines the unknown QUANTITY of the equation. In the conditional form an equation is stated for each possible case. It is that latter form that allows the end-user to declare implicit equations. The analog equations can be GUARDED (see analog discontinuities).
The explicit equations can be realized as CONCURRENT QUANTITY ASSIGNMENT, that can eventually be GUARDED. These relations can update explicit or implicit QUANTITIES (contributions or across of BRANCH and NODE). The equivalent relation is the equation having TARGET as the left hand side of the relation. The possible forms are simple, conditional and selective. The conditional forms do not need to be complete. If a IF statement does not select an equation the INERTIA principle is applied.
The implicit quantity N'CONTRIBUTION is created when the node N is targeted by a Quantity Relation Statement. A node can have several contributions. Each contribution has its own associated analog driver. It is possible to update the current through a node by using relations affecting parallel contributions of the node. When a contribution is the source for a relation the label of the relation where it is computed is required.
The PROCEDURAL form allows to describe a simultaneous statement by grouping analog sequential statements. The PROCEDURAL can be GUARDED.
It is possible to define the functionality of an analog system through the call of a concurrent analog procedure. This call to a subprogram is translated into a call to PROCEDURAL and sequential calls.

Sequential Domain Inside PROCEDURAL statements the QUANTITIES are updated by SEQUENTIAL QUANTITY ASSIGNMENT. These assignments can update explicit and implicit QUANTITIES (contributions or across of BRANCH and NODE). The equivalent form of the relation is the equation having the left side of the relation as TARGET. The possible forms are simple, conditional and selective. The conditional forms do not need to be complete. If the IF statement does not return an equation the INERTIA principle is used.
It is possible to use the NULL sequential statement.
It is possible to define the function of an analog system through the call of an analog procedure.
Any non digital instruction can be used in a procedural block.

OPAL

Concurrent domain The equation has three forms, built around the new == operator. The three possible statements, simple, if and case are presented in the following table. These constructions can be nested.
It is possible to use the NULL simultaneous statement.
The PROCEDURAL form encapsulates sequential statements and provides a way to describe simultaneous behavior through sequential instructions.
The BREAK instruction allows to define new initial conditions when a discontinuity is detected (see analog discontinuities). It is a new "digital instruction", defined for convenience in this section since the analog kernel is synchronized by this expression.

Sequential domain Sequential statements are encapsulated by the procedural statement. This part is scarcely documented.
A sequential variant of the BREAK is also available (see analog discontinuities)

CONCURRENT	**JADE proposal**
==	Equation is represented by exp1 == exp2 Three forms of analog equation statements are defined simple, conditional and selected. [label:](target) => [guarded] expression1 == expression2; [label:](target) => [guarded] equ1 when cond1 else equ2 when cond2 else equ3 when cond3 else equn;

CONCURRENT	**JADE proposal**
	[label:] with exp select (target) => [guarded] equa1 when choice1, equa2 when choice2, equan when others; target can be a n NODE, BRANCH or QUANTITY If it is a NODE, then NODE'CONTRIBUTION is the unknown If it is a BRANCH then BRANCH'THROUGH is the unknown
<=	Quantity assignment [label:] quantity <= [guarded] exp1; [label:] quantity <= [guarded] exp1 when cond1 else exp2 when cond2 else expn when cond else exp [label:] with expression select quantity <= [guarded] exp1 when cond1, exp2 when cond2, expn when cond; If no value is assigned to a quantity, Q'INERTIA is used.
GUARDED	see analog discontinuities
N'CONTRIBUTION (label)	Updating the CONTRIBUTION on a NODE can be achieved by means of a parallel assignment, see example below. The label specifies the equations where this CONTRIBUTION is the unknown
PROCEDURAL	[label:] [GUARDED] PROCEDURAL [is] declarative_part begin analog_and_neutral_sequential_statements end [guarded] procedural [label];
CONCURRENT ANALOG SUBPROGRAM	modeling an analog behavior can be achieved by means of an analog procedure call

Table 7: JADE analog concurrent behavior

SEQUENTIAL	**JADE proposal**
<=	[label:] quantity <= expression;
analog procedure call	

Table 8: JADE analog sequential behavior

CONCURRENT	**OPAL proposal**

==	there are three cases of figure of simultaneous statements simple, if and case
	
	[label:] expression1 == expression2;
	
	[label:] if cond1 use
	simultaneous_statements;
	elseif cond2 use
	simultaneous_statements;
	elseif condn use
	simultaneous_statements;
	else
	simultaneous_statements;
	end use;
	
	[label:] case exp use
	when choice1 => simultaneous_statements,
	when choice2 => simultaneous_statements,
	when choisen => simultaneous_statements;
	end use;
	simultaneous_statements can be simple, if, case or null simultaneous statements.

PROCEDURAL	[label:] PROCEDURAL [is] declarative_part begin sequential_statements end procedural [label]; It is the only grammar rule in the documentation
CONCURRENT ANALOG SUBPROGRAM	not documented

Table 9: OPAL analog concurrent behavior

SEQUENTIAL	**OPAL proposal**
Quantity assignment	Not documented or not available
analog procedure call	not documented
NULL	not available or not documented

Table 10: OPAL analog sequential behavior

SEQUENTIAL	**OPAL proposal**
BREAK	New digital instruction (see analog discontinuities)

Table 11: OPAL New digital instructions

2.6 Analog connexions

VHDL'93 provides generics and ports for interconnecting devices or models in a design hierarchy. Ports provide channels for dynamic communication. The same principle is applied for NODE/TERMINAL and

QUANTITY.
Interface mechanisms are available for design entity, component, block and parameter of subprogram.

PORT	**JADE proposal**
	port(NODE: nature_type := default_exp; QUANTITY: mode analog_type := default_exp); A static QUANTITY can be passed to a subprogram A NODE can not be passed to a subprogram
QUANTITY	A QUANTITY PORT has IN, OUT or INOUT mode (IN is the default). With mode IN, a QUANTITY can be read by the block. With mode OUT, a QUANTITY can be updated by the block. With mode INOUT, a quantity can be read and updated. In an association list QUANTITY PORT can be left unconnected with mode OUT or mode IN with default expression.
NODE	A NODE PORT has no mode. A NODE PORT can be left unconnected if it has a default expression In association list, actual ports and formal ports must represent the same class of object of same type.

Table 12: JADE analog ports

PORT	**OPAL proposal**
	port(TERMINAL: nature_type; QUANTITY: mode real_type := default_exp);
QUANTITY	A QUANTITY PORT has IN, OUT mode (IN is the default). With mode OUT, the QUANTITY PORT provide one characteristic expression, and the QUANTITY may be considered as an output. With mode IN, a QUANTITY acts as an input. The mode of QUANTITY PORTs permits to count characteristics expressions and make a diagnose about the solvability of DAE system. A QUANTITY can be passed as a parameter to a subprogram In association list QUANTITY PORT with mode OUT can be left OPEN.
TERMINAL	A TERMINAL PORT has no mode and no default expression. A TERMINAL PORT can be unconnected. In association list, actual ports and formal ports must represent the same class of object of same types.(see mixed netlist for an extension of this mechanism)

Table 13: OPAL analog ports

2.7 Mixed netlist

JADE A sentence in the proposition states that "... a new mechanism must be implemented that generates nodes or quantities from signals and vice versa."

Actually the "mixed" connexions are handled by the user and are used with the explicit inclusion of some code (DA and AD converters).

OPAL OPAL presents a study about how to interconnect directly signals, terminals and quantities. Two possible solutions are presented: Insertion of explicit or implicit code. Both ways are problematic.
No conclusion has been given.

2.8 Mixed simulation

An abstract VHDL-A simulator is constituted by a digital and an analog kernel. Each kernel manages its own objects. It must be possible to make an analog description sensitive to an event and a digital description sensitive to an analog quantity.
Four cases of mixed simulation can be extracted:
Conversion of object Quantity -> Signal or Signal -> Quantity :

- ENTITY with SIGNAL input and QUANTITY Output
- ENTITY with QUANTITY input and SIGNAL Output

Mixed models :

- The behavior of a quantity depending on the value or the activity of a signal.
- The behavior of a signal depending on the value of a quantity.

Thus it must be possible to read signals (value and events) and the value of quantities anywhere.
In both proposals, QUANTITIES can be read by a process and a SIGNAL can be read by a concurrent statement (analog process).

JADE The simulation kernels are able to see all expressions. Neither an analog quantity can be updated by the digital kernel nor a signal can be updated by the analog solver.
The synchronization of analog behavior through an event can be performed by the following PROCEDURAL statement:

```
 PROCEDURAL IS
  IF real(S´Last_event) = NOW then
     ...
  END IF
  ...
END PROCEDURAL
```

The synchronization of a process through the analog kernel implies the definition of new implicit signals or analog events. These analog events are boolean signals and are issued by the analog kernel. Q'CROSSING(E), Q'FALLING(E) and Q'RISING(E) can be used as synchronization signals in the process environment. The threshold E is a dynamic expression. These attributes can be applied to implicit quantities.

```
PROCESS
  BEGIN
   .....
   wait on Q´CROSSING(E);
   ...
   wait on Q´DDT´RISING(A);
  END;
```

To summarize, no special mechanism is needed to synchronize analog and digital kernels.

OPAL The VHDL kernel updates the driver of the implicit signal Q'ABOVE(E) where Q is a QUANTITY, E a value. Q'ABOVE(E) is a boolean signal available for process sensitivity list. The instruction WAIT ON Q'ABOVE(E); is valid in a process.
OPAL defines the attributes on signals S'NEXT_VALUE and S'NEXT_TIME which hold the time and value of the next transaction on SIGNAL S.
The mechanism to synchronize the analog solver on the digital kernel is implemented by the break statement (see analog discontinuities).

	JADE proposal		**OPAL proposal**
Q'RISING (level)	Set to FALSE at initialization, generates a boolean signal and toggles it when quantity Q crosses the "level" upwardly	N.A.	N.A.
Q'FALLING (level)	generates a boolean signal and toggles it when quantity Q crosses the "level" downwards	N.A.	N.A.
Q'CROSSING (level)	generates a boolean signal and toggles it when quantity Q crosses the "level"	Q'ABOVE (level)	returns TRUE when quantity is above threshold and else FALSE. updated by the digital kernel.
		'Next_value 'Next_time	Value and time of the next transaction on a signal.

Table 14: "analog events"

2.9 Analog discontinuities

There are four cases of analog discontinuities :

1. Event forces reevaluation of a quantity
2. Subprogram incorporates discontinuities
3. Swapping DAE's
4. Instantaneous change in the DAE solution (ex : tunnel diode)

JADE The proposed solutions are :

1. The IF statement with a condition enclosing an event is sufficient.
2. It is impossible to predict what function will cause discontinuities. It is up to the model writer to provide indications.
3. Must be managed by the model writer
4. The most difficult, the model writer must describe the discontinuity

For an efficient handling the analog kernel must know :

- when the discontinuity occur,
- which equation parameters are changed and their new values.

The proposed mechanism is an extension of the GUARDED signal assignment. The GUARDED statements (quantity assignment and equation statement) are used when it is necessary to give new values for a QUANTITY after a discontinuity. The GUARDED statement replaces, momentary, the statements which drive a QUANTITY, NODE or BRANCH when the point after the discontinuity has to be computed. The GUARDED analog statement is only active when the GUARD EXPRESSION is true (cannot be true for more than one ASP). Many guarded statement can drive the same QUANTITY but not at the same time. The GUARDED statements are encapsulated in a GUARDED BLOCK. The associated condition (the guard) is generally associated with an analog generated event like Q'CROSSING(level)'LAST_EVENT.

```
Example : ARCHITECTURE ball OF bouncer IS
            QUANTITY v : analog := 0.0;
            QUANTITY s : analog := 10.0;
            QUANTITY g : analog := 9.81;
            QUANTITY air_res : analog := 0.1;
         BEGIN
           BLOCK (REAL(S´CROSSING(0.0)´Last_event)=NOW)
           BEGIN
              v <= guarded -v
           END BLOCK;
           (s) => s´DDT == v;
           (v) =>    v´DDT == -g - v**2*air_res when v > 0.0
               else  v´DDT == -g + v**2*air_res;
        END BALL;
```

Note: A GUARDED analog statement never targets a node.

OPAL The proposed mechanism by OPAL, for the four cases, is based on the new digital BREAK statement. The sequential BREAK statement is embedded in a conventional VHDL PROCESS. It signals a discontinuity at that time by scheduling a zero-delayed transaction on the implicit break signal and can specify new initial conditions for selected QUANTITIES. The implicit break signal cannot be referenced by the model.

When the solver detects an active break-signal it performs special processing in order to handle the discontinuity. The BREAK statement is not allowed in postponed PROCESS.

"The BREAK statement puts the reponsability for marking discontinuities in the hand of the modelers."

The sequential BREAK :

```
[label:] BREAK [break_list] [WHEN condition];
 break_list ::= break_element {,break_element}
 break_element ::= quantity_name => expression

 Example :  BREAK v => -v when not s´Above(0.0);
```

The condition, if present, is first evaluated. A break is indicated when the condition is true (or no condition), and the driver of the BREAK signal is assigned with the waveform TRUE AFTER 0 SEC and then each break element is evaluated (in the written order). The QUANTITY and the evaluated value are added to the BREAK set for the current simulation cycle.

The concurrent BREAK :

[label:] BREAK [break_list] [ON sensitivity_list] [WHEN condition];

The equivalent process is

```
[label:]PROCESS
        BEGIN
          BREAK [break_list] [WHEN condition];
          WAIT [sensitivity_list]
               |if no sensitivity_list[signals_in_condition];
        END PROCESS;

  Example : ARCHITECTURE ball OF bouncer IS
              QUANTITY v : velocity := 0.0;
```

```
    QUANTITY s : displacement := 10.0;
    QUANTITY g : real := 9.81;
    QUANTITY air_res : real := 0.1;
 BEGIN
   BREAK v => -v ON not(s´above(0.0))
   s´DOT == v;
   IF v > 0.0 use
      v´DOT == -g - v**2*air_res;
   else use
      v´DOT == -g + v**2*air_res;
   end use;
END BALL;
```

2.10 Interpretation Domain (ID)

Both proposals make the same introduction about Interpretation Domains. But, besides general recommendations, no concrete support of these Interpretation Domains is proposed. Especially no support for expressions written in frequency domain is given (Laplace operator **s** or Fourier operators are not supported). From all possible analysis the classical three ones that use SPICE are on the list of "near to-do" (according to the DO4 and DO5):

- the transient ID, to compute any time point
- the DC ID to compute any DC or operating point
- the AC ID to compute any small signal frequency point

For the moment simulation is exclusively performed in temporal domain. The interpretation domain is transient.
Both proposals propose an impure function returning the frequency.

JADE Proposes, until a future release, to use the existing semantic IF analysis_mode THEN.

OPAL Proposes orthogonally to the frequency function an also impure analog_time function. OPAL points out that an overloading of the analog attributes ('DDT, 'INTEG, etc.) should be defined, and attempts to declare a TYPE ANALOG.

3 Syntax and semantic analysis

3.1 Rationale

The JADE presentation is more readable than OPAL's. The underlined concepts are more usable for the reading of the remaining of documentation.

3.2 New type

The COMPLEX new type in the JADE proposal prepares the future work. Currently there is no benefit for this type, but it would be difficult to make frequential domain analyzes without it.
Furthermore thanks to the dimorphism of the ANALOG subtype, change of interpretation domain remains transparent to the user.

3.3 Time definition

The two approach are slightly different but we think they are relatively close from the user's point of view.

3.4 Analog objects

The dimorphic aspect of the analog type of JADE prepares the future work. The REAL type of OPAL seems to be limited to the current proposition.
The NATURE declaration of the OPAL proposal defines a new mechanism and the extension to array and record of nature is less straightforward than the extension of type declaration of JADE.

JADE

```
SUBTYPE voltage IS ANALOG;
SUBTYPE current IS ANALOG;
-- Nature declaration, extension of ARRAY/ACCESS/RECORD
TYPE electric IS NATURE
   ACROSS voltage;
   THROUGH current;
END NATURE electric;
-- A vector of nature, VHDL´93 like type declaration
TYPE elec_array is ARRAY (integer RANGE<>) of electric;
```

OPAL

```
SUBTYPE voltage IS REAL;
SUBTYPE current IS REAL;
-- Nature declaration, new mechanism
NATURE electric IS
   voltage ACROSS
   current THROUGH;
-- A vector of nature, new mechanism
NATURE elec_array is ARRAY (integer RANGE<>) of electric;
```

3.5 Analog behavior

A great part of possibilities announced by OPAL are not presented. The whole part about analog procedural modeling is not available. Nothing is said about the assignment of quantities in sequential domain and about the possibilities of analog procedure calls (concurrent or sequential).
Very powerful analog simultaneous statements are presented in OPAL. The equivalent construction in JADE is the PROCEDURAL in the sequential domain.
The two following examples illustrate the two techniques to design conditional models.

JADE

```
-- ENTITY mos0, ARCHITECTURE one OF mos0
LIBRARY IEEE;
USE IEEE.ASTANDARD.all;
ENTITY mos0 IS
    GENERIC (mtype : mos_type;    -- nmos or pmos
             xg : real := 1.0;    -- transconductance (A/V**2)
             vt : real := 0.0);   -- threshold voltage
    PORT (NODE d, g, s : electrical);
END ENTITY mos0;
ARCHITECTURE one OF mos0 IS
    CONSTANT p : real := REAL(1 - 2*mos_type´pos(mtype)); -- +-1
    BRANCH ds : electrical is d TO s;
BEGIN
    beh : procedural
      VARIABLE ids, vgs, vgd : ANALOG;
    BEGIN
      vgs := g´across - s´across;
      vgd := g´across - d´across;
      IF p*ds´across >= 0.0 THEN            -- forward
        IF p*(vgs - vt) <= 0.0 THEN         -- cut off
            ids := 0.0;
        ELSIF p*(vgs - vt) <= p*ds´across THEN    -- saturation
            ids := p * 0.5 * xg * (vgs - vt)**2;
        ELSE                               -- triode
            ids := p * xg * (vgs - vt - 0.5*ds´across) * ds´across;
        END IF;
      ELSE                          -- reverse
        IF p*(vgd - vt) <= 0.0 THEN         -- cut off
            ids := 0.0;
        ELSIF p*(vgd - vt) <= -p*ds´across THEN   -- saturation
            ids := -p * 0.5 * xg * (vgd - vt)**2;
        ELSE                               -- triode
            ids := p * xg * (vgd - vt + 0.5*ds´across) * ds´across;
        END IF;
      END IF;
      ds´through <= ids;
    END PROCEDURAL beh;
END ARCHITECTURE one;
```

OPAL

```
ENTITY mos0, ARCHITECTURE one OF mos0

USE disciplines.electrical_system.all;
ENTITY mos0 IS
    GENERIC (mtype : mos_type;     -- nmos or pmos
             xg : real := 1.0;     -- transconductance (A/V**2)
             vt : real := 0.0);    -- threshold voltage
    PORT (TERMINAL d, g, s : electrical);
END ENTITY mos0;
ARCHITECTURE one OF mos0 IS
    CONSTANT p : real := REAL(1 - 2*mos_type´pos(mtype)); -- +-1
    QUANTITY vgs ACROSS g TO s;
    QUANTITY vds ACROSS ids THROUGH d TO s;
    QUANTITY vgd ACROSS g TO d;
BEGIN
    IF p*vds >= 0.0 USE            -- forward
        IF p*(vgs - vt) <= 0.0 USE         -- cut off
            ids == 0.0;
        ELSIF p*(vgs - vt) <= p*vds USE    -- saturation
            ids == p * 0.5 * xg * (vgs - vt)**2;
        ELSE                               -- triode
            ids == p * xg * (vgs - vt - 0.5*vds) * vds;
        END USE;
    ELSE                           -- reverse
        IF p*(vgd - vt) <= 0.0 USE         -- cut off
            ids == 0.0;
        ELSIF p*(vgd - vt) <= -p*vds USE   -- saturation
            ids == -p * 0.5 * xg * (vgd - vt)**2;
        ELSE                               -- triode
            ids == p * xg * (vgd - vt + 0.5*vds) * vds;
        END USE;
    END USE;
END ARCHITECTURE one;
```

OPAL does not give indications about the use of the concurrent NULL statement.

We are not convinced about the utility of the TARGET quantity in the JADE equations. For instance:

If we make a model of a resistor in a simple manner:

JADE

```
ENTITY res IS
  GENERIC( r : real :=10 );
  PORT (NODE p,m : electrical);
END;
ARCHITECTURE simple OF res IS
BEGIN
  ([p,m]) => [p,m]´across == r * [p,m]´through;
END;
```

In the JADE proposition it is said that when a branch B is the target then B'through is implicitly the unknown. But if this model is connected to a current source, the unknown would be B'across. So nothing

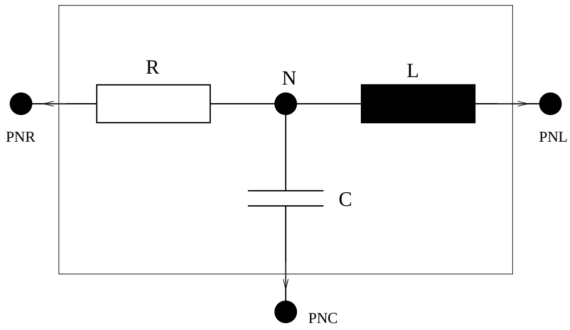

Figure 1: Example of a RLC filter

can be determined by the model writer concerning the future use of its model. Naming TARGETS in the equations does not seem to simplify the problem.
Thus we are not convinced naming TARGETS is mandatory (without particular mathematical study).
The updating mechanism that relies on parallel CONTRIBUTIONS in JADE seems to be very powerful, but at least in the electrical domain it makes the code for models longer and more complex. Furthermore, it is possible to construct non conservative models using CONTRIBUTIONS. In the case of really complex models, we think that the discrepancies due to this mechanism will be really difficult to manage.
The following example (see picture 1) illustrates the modification of the models using explicit BRANCH, implicit BRANCH and the use of CONTRIBUTIONS. For the sake of completeness we have also put the equivalent in OPAL.

```
entity JADE_RLC is
        generic (R, L, C : REAL);
        port (node PNR, PNL, PNC : ELECTRICAL);
end entity JADE_RLC;

architecture with_contributions of JADE_RLC is
        node N : ELECTRICAL;
begin
        l1: N´contribution <= (N´across -PNR´across)/R;
        l2: N´across <= PNL´Across + L * N´contribution´ddt;
        l3: N´contribution <= C * (N´across´ddt - PNC´across´ddt);
        l4: PNR´contribution <= - N´contribution(l1);
        l5: PNL´contribution <= - N´contribution(l2);
        l6: PNC´contribution <= - N´contribution(l3);
end architecture with_contributions

architecture with_explicit_branch of JADE_RLC is
  node N : ELECTRICAL;
  branch br : ELECTRICAL  is PNR to N;
  branch bl : ELECTRICAL  is PNL to N;
  branch bc : ELECTRICAL  is PNC to N;
begin
  l1: br´through <= br´Across / R;
  l2: bl´across <= L * bl´through´ddt;
  l3: bc´through <= C * bc´across´ddt;
end architecture with_explicit_branch
```

```
architecture with_implicit_branch of JADE_RLC is
  node N : ELECTRICAL;
begin
  l1: [N,PNR]´through <= [N,PNR]´Across / R;
  l2: [N,PNL]´across <= L * [N,PNL]´through´ddt;
  l3: [N,PNC]´through <= C * [N,PNC]´across´ddt;
end architecture with_implicit_branch
```

```
entity OPAL_RLC is
  generic (R, L, C : REAL);
  port (terminal PNR, PNL, PNC : ELECTRICAL);
end entity OPAL_RLC;

architecture OPAL of OPAL_RLC is
        terminal N : ELECTRICAL;
  quantity VR across IR through PNR to N;
  quantity VL across IL through PNL to N;
  quantity VC across IC through PNC to N;
begin
        l1: VR == R * IR;
        l2: VL == L * IL´dot;
        l3: IC == C * VC´dot;
end architecture OPAL;
```

Both propositions define for each declared nature N a reference node N'REFERENCE such that the across parameter of the node is stuck to 0.0. Our reflections brought us to ask ourselves the following question. If we model a system using a galvanic isolation (transformator or opto-coupler), the ground potentials on each side of the system do not have a common reference. The question is: How to model such systems and simulate independently of the potential-difference of the grounds?

3.6 Analog connexions

In OPAL the QUANTITIES can only be in mode IN or OUT. In JADE it is also possible to use the mode INOUT.
It appears that the mode of the QUANTITY in OPAL deserves the problem of calculating the number of equations in a model. This functionality is the same as the TARGET in JADE.
In OPAL no work was presented on the interface lists and the call of functions.

3.7 Mixed netlist

The state of completion is far away from complete for both proposals. The theoretical studies of OPAL seem to be more advanced, they present different possibilities of mixed class port connexions, but do not propose any solutions.
JADE proposes to insert explicit code to support the different classes (such as D/A, A/D converters). This method would always work but would also require supplementary work from the user. It could be thought of environments of compilation/verification which include automatically such converting code.
It must be said, that the only way to keep the VHDL philosophy, "the right architecture for the right analysis" can only be kept if the connexions are generalized. A logical architecture should be replaceable by an analog

one without rebuilding the structural definition. For the future enhancement such a mechanism must be proposed.
At last, we think that the DO33 and DO36 were interpreted in a too large sense. There is no requirement concerning the connexion of SIGNAL and TERMINAL.

3.8 Mixed simulation and Analog discontinuities

The analog events (RISING,FALLING,CROSSING) proposed by JADE are more powerful and allow interfacing with hysteresis directly. The GUARDED mechanism is an extension of the VHDL'93 but there are two drawbacks. First, the realization of the GUARD condition depends on the equality of two reals (hard to compute). Second, to ensure correct time-resolution, a backtrack algorithm should be associated to the condition evaluation between the (analog) NOW function based on ASP and the time of the event realization. Is the analog event stamped with analog or digital time ? If for instance, the Q'CROSSING(level) occurs between two ASP, the analog kernel has to retrieve the time corresponding to the actual crossing between Q and level.
In OPAL we do not understand the meaning of a BREAK statement without the WHEN clause. Is it always true ?
A concurrent BREAK statement can exist without WHEN and ON clauses referring to the grammar rule in the OPAL proposal. The consequence is an equivalent process with any wait statement. It is not conformant to the VHDL'93 law (see 2.1 in OPAL's MIXED MODE SIMULATION document).
The JADE proposal implements the analog/digital interaction in a more straightforward manner :

JADE

```
LIBRARY IEEE;
USE IEEE.ASTANDARD.all;
ENTITY vpwl IS
    GENERIC (v0 : real := 0.0);
    PORT (SIGNAL slope : in real;
          NODE p, m : electrical);
END ENTITY vpwl;


-- ARCHITECTURE one OF vpwl

ARCHITECTURE one OF vpwl IS
    BRANCH pm : electrical is p TO m := v0;
BEGIN
    (pm) => pm´across´ddt == slope;
END ARCHITECTURE one;
```

OPAL

```
ENTITY vpwl

USE disciplines.electrical_system.all;
ENTITY vpwl IS
    GENERIC (v0 : voltage := 0.0);
    PORT (SIGNAL slope : in real;
          TERMINAL p, m : electrical);
END ENTITY vpwl;
```

```
ARCHITECTURE one OF vpwl

ARCHITECTURE one OF vpwl IS
    QUANTITY v := v0 ACROSS i THROUGH p TO m;
BEGIN
    BREAK WHEN slope´event;
    v´dot == slope;
END ARCHITECTURE one;
```

3.9 Interpretation Domain

In both cases work do not show much progress. JADE proposes to use explicitly IF blocks of sequential domain: IF analysis_domain THEN, in order to support the different descriptions and, even if JADE does not mention it, it is also possible to perform this operation in the concurrent domain using the IF analysis_domain USE construct.

4 Completeness of presentation

4.1 Quality of documentation

Both documents are well written and quite pedagogic, but the documents of the JADE proposal are easier and more readable. The LRM project allows a more formal presentation and is better organized regarding a certain number of concepts.
On the other hand the OPAL proposal presents lots of examples that illustrate precisely the proposed concepts. The OPAL documents hold too much "to be done...", "no conclusion" and too much omissions about important points.
The documentation "work to be done" in order to transform one of the propositions in definitive documentation is in favor of JADE.

4.2 Adherence to VHDL-A Design Objective Document (DOD) and Design Guidelines

The LDC design guidelines are the following:

- Upward compatibility
- Preserve strong typing
- Unification of timing semantics
- Preserve determinism
- Preserve generality
- Preserve scope of VHDL (gate to system)
- Preserve intermixed abstraction levels
- Preserve concurrency
- Preserve and improve consistency
- Preserve and improve portability
- No application specific packages
- Minimize implementation impact
- Maximize implementation efficiency
- Distinguish between the language, modeling and tool issues/aspects.
- Ease of use[4].
- Constructive rather than propositional definitions

[4]see General recommendation

These guidelines are followed by the two proposals.
The DOD's are not represented in this document. They can be retrieved on the www.vhdl.org WEB server.

MUST	**JADE proposal**	**OPAL proposal**
DO0	OK	OK
DO1	OK (1)	OK (1)
DO3	OK	OK
DO4	OK (2)	OK (2)
DO5	OK	NATURE/BREAK (3)
DO7	NO (4)	NO (4)
DO9	NO (5)	NO (5)
DO10	OK (6)	OK (6)
DO11	OK	OK
DO13	OK	OK
DO14	OK	OK
DO18	OK	partially (7)
DO19	OK	? (8)
DO20	OK	OK
DO21	OK (13)	OK (13)
DO22	OK	OK
DO23	OK (9)	OK (9)
DO24	OK	OK
DO25	OK	OK
DO26	OK	OK
DO27	NO (10)	NO (10)
DO33	OK	OK
DO34	OK	OK
DO35	OK	OK
DO37	OK	OK
DO38	OK	OK
DO39	OK	OK

Table 15: Conformance to "must" design objectives

SHOULD	**JADE proposal**	**OPAL proposal**
DO2	OK	OK
DO6	OK	OK
DO8	NO (4)	NO (4)
DO9b	OK	OK
DO12	OK (11)	OK (11)
DO15	NO (12)	NO (12)
DO28	OK	OK
DO29	OK	OK
DO30	NO	NO
DO32	NO	NO
DO36	OK	OK
DO40	OK	OK

Table 16: Conformance to design "should" objectives

DESIRABLE	JADE proposal	OPAL proposal
DO7b	NO	NO
DO8b	NO	NO
DO16	NO	NO
DO17	OK	OK
DO31	NO	NO

Table 17: Desirable design objectives conformance

(1) See general recommendation (2) It is the only domain (3) See general conformance (4) By user hand (5) By user hand (6) See mixed simulation (7) Affectation not documented (8) Sequential part not documented (9) Only the time (10) Reported on standard math package (11) By explicit code only (12) Not yet explicated (13) Misinterpreted, see General recommendation

4.3 General conformance to VHDL'93

Generally the conformance to VHDL'93 is respected by the two proposals.
It is regrettable that OPAL introduces a new syntax for the new mechanisms (NATURE, BREAK), while JADE tries to associate them to the existing syntax (TYPE ..IS NATURE, GUARDED).
The general philosophy of VHDL'93 is respected. The descriptions will be compatible in the limits imposed by the recommendations.
Very few comments are made concerning the SHARED variables. It is probable that their use will cause problems (as in VHDL'93).
Further the quite intensive use of impure functions in JADE can lead to non portable models, enhancing on the other side considerably consistency to existing VHDL semantic and flexibility.

4.4 Status of completion

The work on frequency domain is far from being completed.
After the previous section, we think that JADE proposal is more advanced in completion than OPAL.

5 The user point of view

5.1 Code length

More or less, the models with JADE or OPAL have comparable code length. With JADE the CONTRIBUTION mechanism has a bad influence on the code length.
We have no examples matching industrial complexity to give a valuable opinion.

5.2 Understanding the code

Like a model written in VHDL'93, the ability to understand the code easily ··· depends more on the coder than on the language itself.
Both propositions allow to write readable and comprehensible code. As in other languages the same methods can be used to produce unreadable code.
Anyway, there are some limitations :

JADE The models based on CONTRIBUTION are difficult to read.
The proposed mechanisms impose the extensive use of attributes with very long names. Therefore JADE proposes shorter aliases, allowing a more coherent analysis of the equations. An expression like `B'ACROSS'DDT == R'CONTRIBUTION(lbl)` would become `B'V'DDT == R'I(lbl)`, supposing that B and R are of type ELECTRICAL. These aliases are declared at the same time as the nature, for example:

```
TYPE electric IS NATURE
  ACROSS V: analog;
  THROUGH I:analog;
END NATURE electric;
```

Some constructions show a lack of elegance, and expressions like `real(s´crossing(0.0)´last_event)=NOW` should be replaced by something like `s´crossing(0.0)´analog_event=NOW`

OPAL The QUANTITY hides two concepts (branch and free) (See general recommendations below)

5.3 Modeling multi level descriptions

There are two drawbacks against the ability to track the multilevel modeling. The problems of non anonymous ports and the ports of QUANTITY class. This problem arises for the first time in VHDL, since, since now, only one element (signal) could be present in a interface list.
The first problem is unresolved in the current propositions.
The support of QUANTITY (with IN or INOUT modes) in the interface list of entity is an open door to non manageable multi-level abstraction. (see General recommendations)

6 Flexibility - future needs

The actual work has not progressed far enough to be able to speak about future needs.
Nevertheless, a mechanism for translation or direct integration of SPICE net-lists must be proposed. The volume of knowledge accumulated in this domain is too important to be ignored. We think that a great effort should be done on this point.
A mechanism to support frequential analysis must absolutely be found. JADE prepares the future with the dimorphic type ANALOG. But this definition is not enough alone.
It can be thought of the future of the integrated circuitry integrating non classical concepts as dispositives with hyper small geometry, quantum wells, optical sub circuits or interconnexions.... The actual state of the propositions for VHDL-A is very far away from such a point of view. Tools allowing to mix the different "analog" approaches and physical phenomena are needed, as the then surely needed support for partial differential equations. A merging effort with MHDL standard could then be thought of.
For the classical electronics, connected domains or transversal disciplines obeying to the same principles we think that VHDL-A is on the right way.

7 General recommendations

To let the user the possibility to make connexions of type QUANTITY seems very limitating. We think that this feature limits considerably the possibility of multiple abstraction levels. It is to be said that no example is given on that theme. If a user has chosen QUANTITIES for his signal flow model, he must modify his structural description to be able to change abstraction model. We recommend that the QUANTITIES accessible in the port list can only be of mode readable.

The simultaneous statements of OPAL are very powerful for the concurrent description of analog behavior and seem to be well adapted to the dynamic adaption of models. We think that the final proposition should incorporate these mechanisms.

In OPAL the word QUANTITY hides two different objects (the free QUANTITY and the BRANCH QUANTITY). To increase ergonomy and easiness, we recommend that the following two keywords are kept (as in JADE): BRANCH for branches and QUANTITY for free QUANTITIES.

The implicit BRANCH declarations of JADE are very easy to use, shorten the models and enhance significantly the readability. This mechanism should be kept in the final proposition.

The mechanism of CONTRIBUTION supplied by JADE seems very powerful, but we think also very dangerous. It can make models unreadable and non conservative. It should be associated with a mechanism of model consistency verification. A BRANCH approach should be kept if possible.

The objects NODE/TERMINAL hide two aspects of reality. They represent the point of connexion of branches and the point where Kirchhoffs law's are respected. We propose to separate these concepts and to name NODE the point where Kirchhoffs law's are respected and TERMINAL the effective terminal of a connexion (in other words: a component that is developed "per se", for future use in a circuit, does not respect Kirchoff).

Only when the TERMINALS are connected to a NODE after elaboration they participate to functionality. A TERMINAL can thus not be left unconnected, but must be attached to a unique NODE.

`HERVE and BOETTCHER proposition`

```
-- WARNING :  NOR JADE NOR OPAL
-- WARNING : only an example, not a true model

ENTITY test IS END;

ARCHITECTURE example OF test IS
 NODE N1,N2 : electric; -- Kirchoff points
BEGIN
  label1: compo1 port map(N1,N2);
  label2: compo2 port map(N1,N2);
END example;

-- The instantiated component
ENTITY compo1 IS
 port(TERMINAL p,m:electric);
END;

ARCHITECTURE simple OF compo1 IS
  NODE N_local: electric; -- Kirchoff point
  BRANCH B_internal is p to N_local;
BEGIN
  (B_internal) =>
       B_internal´across == B_internal´through * 3.0;
  ([p,m]) =>
       [p,m]´across == B_internal´across * 2.0;
END simple;
```

This definition simplifies the approach, the associated explanations and the pedagogy towards the end-user. Here the notions of TOP NODE or ROOT TERMINAL become obsolete.

8 Conclusion

This study shows the great amount of work realized by the teams who prepared the two proposals.
The two proposals are of good quality and present few weaknesses and lot of strengths.

Plus for OPAL:

- Powerful simultaneous statements.

- Lots of examples.

Minus for OPAL:

- Announce of lots of concepts without implementation: no procedural, sequential assignment of quantities, analog procedure calls, interface lists or function calls.
- Declaration of new mechanisms: NATURE, BREAK.
- QUANTITY hides two concepts: branch and free QUANTITY.
- less compliant to DOD than JADE proposal.

Plus for JADE:

- Better structured proposal, thanks LRM project ease of finding punctual concepts.
- Extension of existing VHDL mechanisms: NATURE extension of TYPE, discontinuities handled by GUARDED.
- Implicit BRANCH are a powerful notation.
- Better analog events. E.g. hysteresis is easier to implement.
- definition of a type COMPLEX for frequency domain analyzes and of a dimorphic subtype ANALOG.
- Alias mechanism shortens considerably the very long names.

Minus for JADE:

- The constraint of putting Targets on concurrent equations is not convincing.
- CONTRIBUTION mechanism is powerful, but has the big problem of leading to non conservative models.
- Usage of mode applied on quantities is unclear.
- GUARDED mechanism is based on equality of two floating point values.

Nevertheless, after the previous analysis, we personally think that the JADE proposal is the most achieved at this time and we recommend that JADE will be the substrate for the future VHDL-A standard of the IEEE.

A Annex

List of Tables

1	New Types	3
2	Definition of time	5
3	Nature declaration and attributes	6
4	Quantity	6
5	Node/Terminal declaration and attributes	7
6	Other implicit quantities	7
7	JADE analog concurrent behavior	9
8	JADE analog sequential behavior	9
9	OPAL analog concurrent behavior	10
10	OPAL analog sequential behavior	10
11	OPAL New digital instructions	10
12	JADE analog ports	11
13	OPAL analog ports	11
14	"analog events"	13
15	Conformance to "must" design objectives	24
16	Conformance to design "should" objectives	24
17	Desirable design objectives conformance	25

B Used documents

Origin: http://vhdl.org
ftp nestor.epfl.ch /pub/vhdl/standards/ieee/1076.1/lang_design/proposals
Opal_ps.tar.gz
Jade_ps.tar.gz
date: 30 01 1996

B.1 OPAL

home.ps Opal Home Page (read it first)
arch_toc.ps VHDL-A Language Architecture (table of content)
arch.ps VHDL-A Language Architecture
quantities.ps Quantities and Terminals
quantities_toc.ps Quantities and Terminals (table of content)
quantities_idx.ps Quantities and Terminals (index)
mmsimul_toc.ps Mixed-Mode Simulation (table of content)
mmsimul.ps Mixed-Mode Simulation
time_toc.ps Time in VHDL-A (table of content)
time.ps Time in VHDL-A
simultstmts_toc.ps Simultaneous Statements (table of content)
simultstmts.ps Simultaneous Statements
initdcsim_toc.ps Initialization and DC Simul. in VHDL-A (table of content)
initdcsim.ps Initialization and DC Simulation in VHDL-A
solvability_toc.ps Solvability of DAE Systems (table of content)
solvability.ps Solvability of DAE Systems
netlist_toc.ps Mixed Net-lists (table of content)
netlist.ps Mixed Net-lists
statmod_toc.ps Statistical Modeling (table of content)
statmod.ps Statistical Modeling
stdenv.txt VHDL 1076.1 Environment
spice.txt SPICE Related Issues in VHDL-A
simctrl_toc.ps Simulation Control (table of content)
simctrl.ps Simulation Control

freq.txt Frequency Domain
allexamples_toc.ps Cumulative Index of Examples for VHDL-A
examples_toc.ps VHDL-A Examples (table of content)
examples.ps VHDL-A Examples
colleagues.ps Acknowledgments

B.2 JADE

VHDL Rationale
ANALOG Rationale
ANALOG LRM