Information Models of VHDL

Cristian A Giumale
Department of Computer Science
Technical University of Bucharest
313 Splaiul Independentei, Bucharest 77206, Romania

Hilary J Kahn
Department of Computer Science
University of Manchester
Oxford Road, Manchester M13 9PL, UK

Abstract – The paper discusses issues related to the application of information modelling to the field of Electronic CAD, using VHDL as the basis for discussion. It is shown that an information model of VHDL provides a coherent and uniform description of the VHDL objects at different levels of the language and of the transformations that interrelate these levels. In addition, it captures the time-dependent aspects of the language. Hence, a hierarchy of VHDL information models can exist which encompasses the range from abstraction to detail and can help support CAD applications in a direct manner.

I. Introduction

A major domain of research in the field of electronic computer-aided design (ECAD) is that of modelling. The aim is to build clear and unambiguous descriptions of the semantics of various ECAD languages and artifacts. Of the many possible modelling methods, information modelling has received particular attention in recent years. For example, the specification of the Electronic Design Interchange Format [5] and that of the Design Representation Programming Interface of the CAD Framework Initiative [3] are accompanied by information models. These models use the specification language EXPRESS [6].

The relatively restricted circulation and application of these early information models and the relative novelty of the field has limited the general acceptance of information modelling as a useful tool with a wide range of applicability. The consideration of information modelling as a competitive alternative to other modelling methods has also suffered because, misguidedly, information modelling is seen only as a means of defining the contents of a data base rather than as a conceptual description of the modelled universe of discourse (**UoD**).

The aim of this paper is to outline some of the specific features which an information model and, in particular, an EXPRESS model, can provide. Although the discussion focusses on VHDL [12], a typical language used in the ECAD field, its relevance is more general. The paper tries to show that (a) an information model of VHDL can integrate different perspectives of VHDL objects directly and uniformly; (b) the model can describe the mappings between these perspectives and (c) the model can keep track of the described perspectives for each language object. These features, taken together, can seldom be found in other kinds of model such as formal models [11], which mainly describe functional aspects of VHDL such as its behavioural semantics, or implementation models [13], which concentrate on the representation of the language objects as data structures.

The problems discussed result from work supported by the UK Defence Research Agency and by the ESIP (ESPRIT 8370) project which aim to produce a comprehensive information model of VHDL'87 and, eventually, of VHDL'93. As the work progressed it become clear that a single model of VHDL is not a satisfactory option from the end user point of view. Instead, a hierarchy of models, each of which describes the relevant aspects related to a specific application and from a specific perspective of VHDL, is a more realistic solution. However, the model at the apex of this hierarchy describes the essential objects and the semantics of VHDL at the design description, analysis, elaboration and simulation levels of the language. It can be seen as an abstraction of all the other models in the hierarchy. Such a model, identified here as the **core** model, is currently under review [8]. It is the basis for this paper.

The discussion which follows applies to VHDL'87, called VHDL hereafter. Note that for brevity the examples in the paper are not meant to be complete.

II. Information Modelling

In contrast to computation, which aims to show how values which characterise a **UoD** are obtained, information modelling is intended to capture the conceptual structure of the **UoD**: the underlying objects and concepts, their relationships and constraints. The **UoD** here is VHDL. The main thrust of computation is to solve; the goal of information modelling is to describe.

The main construct of a VHDL information model is the description of a VHDL object. It specifies the **attributes** of the object and the **constraints** which the values of the at-

32nd ACM/IEEE Design Automation Conference ®

tributes must satisfy. For example, using EXPRESS, the object **explicit_signal** which designates an explicitly declared signal, can be described by means of a construct, called **ENTITY**, as in example 1. Note that here **attribute**, **constraint** and **ENTITY** are EXPRESS terms.

This partial example illustrates the main points of an object description. Each attribute has a name and a type, which is the name of another entity. It specifies that in an entity instance the value of the attribute is normally an instance of the entity which represents the type of the attribute. For example **disconnection_delay** is an attribute the values of which are of type **time_expression**, itself an entity which must be fully specified in the model. The attributes in the example are: mandatory (e.g. **signal_type**, inherited from the **signal** entity), optional (e.g. **disconnection_delay**) and computed (e.g. **is_guarded**).

A constraint specifies either a local condition on the values of an entity attribute or a relationship between the values of the attributes belonging to the same entity or to different entities. It is a logical expression which must not evaluate to false for each valid instance of the containing entity. For example, the constraint **valid_disconnection** states that each signal (in reality each instance of the entity **explicit_signal**) can be associated with a disconnection delay only if it is a guarded signal, i.e. if the attribute **signal_kind** is explicitly specified. Predefined or user defined functions can be called within constraint expressions. In the example above **is_type** and **subelement_of_resolved_signal** are user defined functions.

Entities can be structured into hierarchies of supertypes--subtypes which can be used for classification purposes and for attribute and constraint inheritance. Entities are grouped to form a **SCHEMA**, which is a sub-model of a specific part of the modelled **UoD**. Entities from one schema can be used in other schemas. In this way a complete model can be partitioned into smaller and conceptually consistent parts.

An EXPRESS model can be represented diagrammatically by using a graphical notation called EXPRESS-G. The diagrams consist of symbols with different formats for different EXPRESS constructs such as entities and types. For example, an entity is represented by a solid rectangle which is starred if the entity contains constraints. The symbols are connected by lines: thin lines connect attributes to their containing entities; thick lines indicate subtype / supertype relationships. Attribute optionality is represented by dashed lines. The directionality of the relationships is indicated by a circle at the target end of a line. Additional information is displayed on thin lines indicating the nature of the attribute, e.g., DER (in Fig. 1) stands for a computed attribute. For instance, the signal model can be represented as illustrated in Fig. 1.

```
SCHEMA design_description_schema;
REFERENCE FROM vhdl_type_schema;
...
ENTITY signal ABSTRACT SUPERTYPE OF
        (ONEOF(explicit_signal,...);
   signal_type:         VHDL_type;
END_ENTITY;

ENTITY explicit_signal ABSTRACT SUPERTYPE OF
        (ONEOF(port,internal_signal) AND
        ONEOF(scalar_signal,composite_signal) AND
        ONEOF(signal_subelement,complete_signal))
        SUBTYPE OF(signal);
-- declared attributes
   signal_kind:         OPTIONAL bus_or_register;
   disconnection_delay: OPTIONAL time_expression;
   resolved_by:         OPTIONAL resolution_function;
DERIVE -- computed attributes
   is_guarded:  LOGICAL:= EXISTS(signal_kind);
   is_resolved: LOGICAL:= EXISTS(resolved_by) AND
        NOT subelement_of_resolved_signal(SELF);
(* A signal is resolved if it is not a subelement
   of a resolved signal and there exists an associated
   resolution function *)
WHERE -- constraints
 valid_signal_type:
   NOT is_type(signal_type,['file_type','access_type']);
 valid_disconnection:
 (* Only guarded signals can have disconnection delays *)
    is_guarded OR NOT EXISTS(disconnection_delay);
END_ENTITY;
END_SCHEMA;
```

Example 1: The partial model of a signal

III. Intra-level Modelling

The basic goal of an information model is to describe the objects of the given **UoD**, outlining their structure, intrinsic properties and direct relationships with other objects of the **UoD**. For instance, a VHDL core model must describe the basic VHDL objects at different levels of the language: source design description (which corresponds to a source VHDL program), design analysis (which considers specific VHDL objects as library units), design elaboration (which transforms an analysed VHDL description into a network of processes controlled by signals), and design simulation (which highlights the time-dependent behaviour of an elaborated VHDL description). There is a sub-model corresponding to each VHDL level and, at each level, the model of an object em-

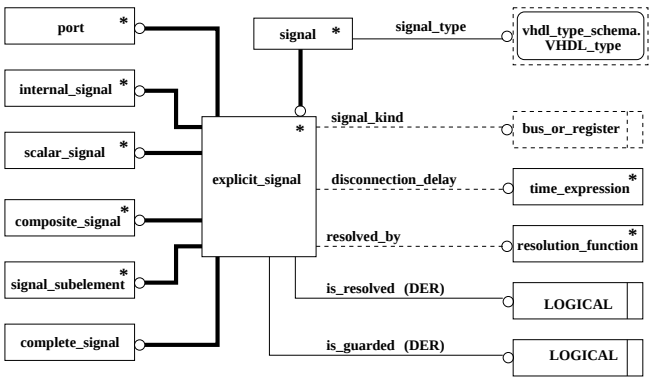

Fig. 1: The diagrammatic representation of the signal model

phasises the possible perspectives of the object.

The **explicit_signal** entity in example 1 illustrates intra-level modelling at the level of source design description. The entity shows that a signal can be classified according to: (a) its role - port or internal_signal (declared in an entity declaration, package declaration, block or architecture), (b) its structure - scalar or composite, and (c) signal membership - subelement of another signal or stand alone signal. Therefore, an **explicit_signal** is an aggregate of these three perspectives. The model outlines the intrinsic properties of an **explicit_signal** by showing that it has a type and may have a kind, an associated resolution function and a disconnection delay. Some properties of a signal depend on other properties. For example, the derived attribute **is_resolved** shows that if the signal is a sub-element of a resolved signal then it is considered unresolved, regardless of the existence of an associated resolution function, i.e., the resolution function plays no role in computing the value of the signal (Std 1076-1987 page 2-7).

Information modelling makes it possible to hide or enhance detail as needed, according to the purpose of the model. The fact that a resolution function is associated with a signal via the subtype of the signal specified in a signal declaration is not described by the signal model nor does the model show that a disconnection delay is associated with a signal via a disconnection specification. These are considered inessential details that are omitted from the core model, the aim of which is to describe only the basic concepts of the language.

IV. Multi-level Modelling

A given VHDL object can have correspondents at different levels of the language. For example, a signal occurs at the source design description and elaboration levels. Each level uses a particular perspective of the object, which implicitly means that the object may have different (but interrelated) descriptions at different levels. Therefore, the abstract model of the object can be seen as the tuple of all its corresponding level-models.

Example 2 illustrates the model of a signal corresponding to the elaboration level of VHDL. The **elaborated_signal** is the entity describing the result of elaborating a **signal**. It is contained in the **design elaboration schema**, which is the sub-model corresponding to the VHDL design elaboration level. The **signal** entity is the entity shown in section II. and specified in the **design description schema** which is the sub-model corresponding to the VHDL design description level.

Each elaborated signal from an elaborated VHDL description corresponds to a source signal from the source description from which it is elaborated. The model shows that there can be several instances of the entity **elaborated_signal** that have the same value for the attribute **source_signal**. In other words, the generic relationship **signal - elaborated_signal** can be one-to-many. In addition, the model from example 2 shows that the nature of a signal (e.g., whether it is a **port** or an **internal_signal**) is no longer important at the elaboration level, where a design is seen as a flat structure containing processes connected by a network of signals [10]. Instead, an elaborated **explicit_signal** is characterised by a set of sources which are used to compute the values of the elaborated signal while the simulation process unfolds.

Based on the sources of a signal, the concept of signal resolution can now be fully described. The constraint **valid_resolution** of the entity **elaborated_explicit_signal** shows that an elaborated signal which has multiple sources and is not a sub-element of a resolved signal of a composite type must be elaborated from a **source_signal** which is associated with a resolution function (Std 1076-1987 page 2-7,4-6).

Note that the model of a VHDL object, such as the model of an elaborated signal, concentrates information that, in the language reference manual (**LRM**), is scattered in different places. In addition, the structuring of the model according to the levels of the language clarifies the abstraction level of the different perspectives of VHDL objects. For example, at the description level, a signal is an abstract object. It plays the role of a template, which cannot have sources, states and values. At the elaboration level a signal has a material existence. An elaborated signal has sources and can have states and values. An information model of a VHDL object and, in general of VHDL, acts both as a concentrator and as a structured representation of the essential information which conveys the semantics of the language.

V. Cross-level Relationships

A major goal of a VHDL core model is to describe the relationships between the objects and object perspectives corresponding to different levels of the language. A typical case

```
SCHEMA design_elaboration_schema;
REFERENCE FROM design_description_schema;
...
ENTITY elaborated_signal ABSTRACT SUPERTYPE OF
          (ONEOF(elaborated_explicit_signal,...));
   source_signal: signal;
END_ENTITY;

ENTITY elaborated_explicit_signal SUBTYPE OF
          (elaborated_signal);
   sources: SET OF elaborated_signal_source;
DERIVE
   has_multiple_sources: LOGICAL:= SIZEOF(sources) > 1;
   has_resolution_function:
      LOGICAL:= EXISTS(source_signal.resolved_by);
   is_subelement_of_resolved_signal:
      LOGICAL:= subelement_of_resolved_signal(source_signal);
WHERE
   valid_resolution:
       ( has_multiple_sources AND
         NOT is_subelement_of_resolved_signal AND
         has_resolution_function) OR
       NOT has_multiple_sources OR
       is_subelement_of_resolved_signal;
END_ENTITY;
END_SCHEMA;
```

Example 2: The partial model of an elaborated signal

is that of describing the salient properties of the algorithmic processes used to generate or transform objects from one language level into objects of another level. These properties are part of the language semantics.

Fig. 2 illustrates the simpler case of signal driver elaboration. A signal driver of a target signal **S** is the set of all the signal assignments the target of which is **S** and which are contained in the same process. All these assignments taken together constitute one potential source of **S**. Depending on the design level being considered, there are: **signal_driver**s, as specified at the source description level of design, and **elaborated_signal_driver**s which result from the elaboration of the **signal_driver**s. Therefore, the entity **signal_driver** is defined in the **design_description** sub-model of the VHDL core model, whereas the entity **elaborated_signal_ driver** is a member of the **design_elaboration** sub-model.

Consider that the entity **elaborated_signal_driver** has the attribute **source_signal_driver**, which designates the **signal_driver** which is elaborated, and contains constraints which show that: (a) there is a one-to-one correspondence between the **elaborated_signal_assignments** of the **elaborated_signal_driver** and the **signal_assignments** of the **source_signal_driver**, and (b) that the **elaborated_signal_assignments** of the **elaborated_signal_ driver** must be contained in the same **elaborated_process**. In other words, the elaboration of a **signal_driver** implies the elaboration of each signal assignment of the driver.

These constraints implicitly specify properties of the elaboration function **signal_driver -> elaborated_signal_driver**, as illustrated in Fig. 3. They show that an **elaborated_signal _driver** is isomorphic to the corresponding **source_signal_driver**. This isomorphism is an example of a cross-level relationship between the design description level and the elaboration level of VHDL. In a similar way, the model can describe other cross-level relationships, such as:

design description | – – analysis —→ design libraries
elaborated design | – – simulation —→ signal process states

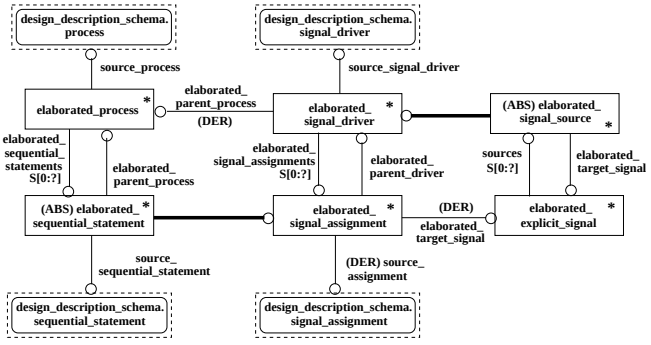

Fig. 2: The model of an elaborated signal driver

VI. Modelling of Time-dependent Behaviour

Intra-level and multi-level modelling of a given VHDL object focus typically on properties that are not time-dependent. The description of behavioural relationships, however, has to cope with time dependent functional correspondences. For example, an important part of VHDL semantics corresponds to the behaviour of a design during simulation. The behavioural semantics address relationships between signals and processes as a function of time. The general model illustrated in Fig. 4 suggests how VHDL behavioural relationships can be modelled.

The attribute **simulation_cycles** of the entity **design_simulation** implicitly specifies the simulation function as a set of points **(previous simulation cycle, current simulation cycle)**, where the entity **simulation_cycle** contains as attributes the set of process and signal states corresponding to the cycle. The points of the simulation function must correspond to valid transitions between the process states. Therefore, additional properties are specified to further constrain the points of the simulation function. These 'behavioural' constraints show how process states relate to signal states via a hierarchy of other entities which include: executed wait statements, executed signal assignments, signal drivers and transactions.

It should be noted that the 'behavioural' constraints do not describe the simulation framework of VHDL, e.g. the kernel process and the different phases of a hypothetical VHDL simulator. Instead they make sure that the functional relationships, with regard to time, between different simulation events are correct. From this point of view the level of abstraction of the information model is higher than the level of abstraction of other VHDL models which describe the behavioural semantics of VHDL indirectly, by considering the semantics of a hypothetical simulation machine [2].

The task of modelling the VHDL behavioural semantics is

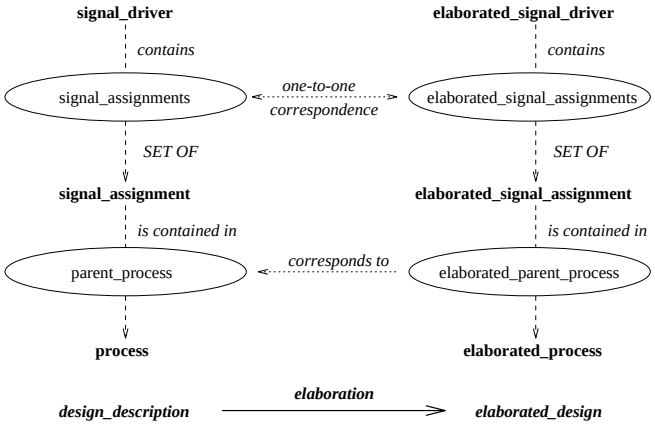

Fig. 3: Cross level modelling

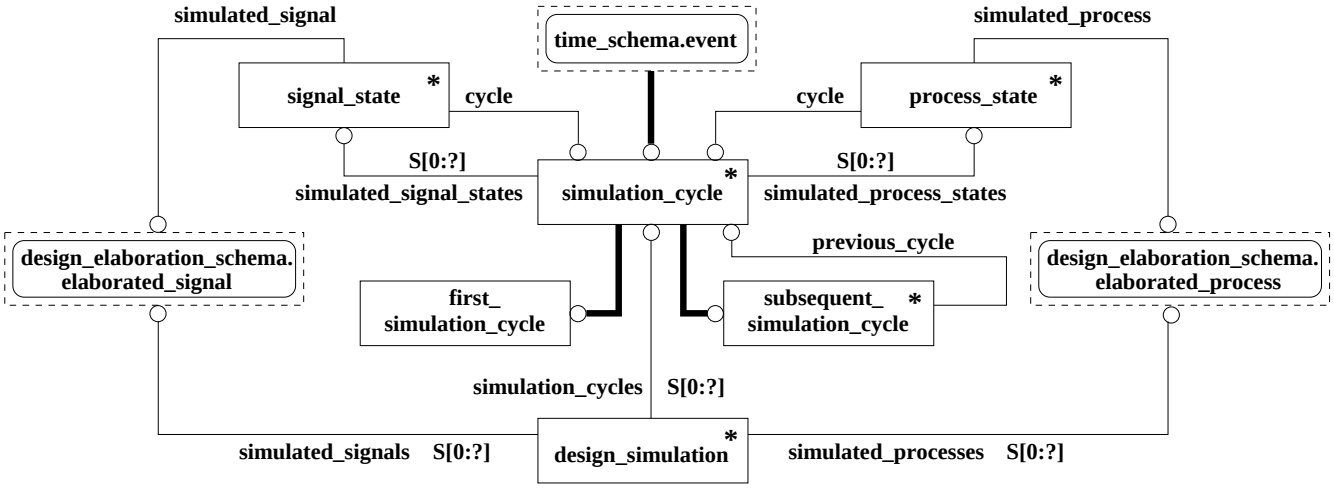

Fig. 4: The general model of the simulation process

slightly more difficult than suggested here. This is due to the VHDL feature of allowing delta delays which means that different object states corresponding to the same object and separated by a null time lapse may co-exist. The model has to be able to order such states as explained in [10]. This discussion is beyond the scope of the paper. A possible solution is presented in [7].

VII. Information Tracking

Besides intra-level, multi-level and time-dependent modelling, an important but sometimes overlooked purpose of a VHDL model is to provide an implementation basis for different applications of the language. To qualify for such a purpose, a model must have two important properties. First the model must have the capability to be instanced. Second, the model must be able to keep track of all the information relevant to the given application. The tracking of information is particularly important since it links a specific result of the application program to those parts of the source design description which are relevant to the result. Generally, any model can keep track of information. The key issue is the balance between the information detail required and retrieval efficiency.

For example, a core model of VHDL could be used to guide the execution of a simulator. Apart from specific behavioural relationships, computing the value of a signal cannot be performed without accessing information such as: signal kind, disconnection delay, associated resolution function. Such intrinsic properties are used by the application control engine to filter the model objects and the relationships which are relevant for the application and which therefore should be processed. This information is gathered selectively from the models of VHDL design description, analysis and elaboration and from their instances according to the detail required by the execution of the application program, as illustrated in Fig. 5.

A single model cannot keep track efficiently of the vast amount of information necessary for a non trivial application of VHDL. It is too costly and impractical. This suggests that efficient information tracking requires the ability to integrate models of different perspectives of the language. This is a natural property of information modelling and an advantage over other kinds of model.

VIII. Conclusions

The view of information modelling as discussed in this paper is somewhat different from the usual perspective which considers an information model as the contents of a data base. The stress on modelling is on conceptual issues. There are important consequences of this idea.

The information modelling of VHDL encourages the stratification of the model according to the level of abstraction and to the roles of the different objects and concepts of the lan-

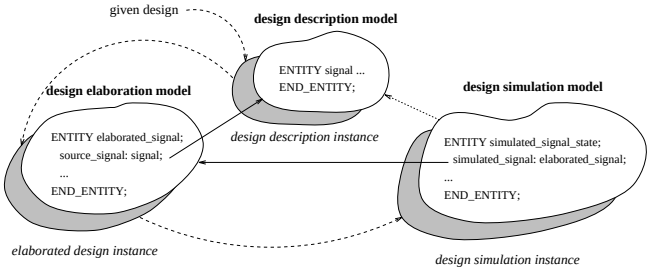

Fig. 5: Information tracking

guage. The paper has shown that static, structural properties, object transformation and dynamic aspects of the language can be appropriately integrated in an information model.

Several information models can exist for VHDL according to the modelling purpose. In particular, it is thought that a core model, seen as the root of a hierarchy of specialised models, can be used to enhance the standard of the language by clarifying the ambiguous aspects and, in addition, as a conceptual comparison base between existing versions of VHDL.

An information model can be taken as a formal specification of the language and, therefore, properties of the language can be inferred from the model. Furthermore, an information model can provide a unique and homogeneous implementation base for application systems. Assuming that an information model forms the conceptual layer of the procedural description of an application [1], and therefore it can be seen as the conceptual layer of a knowledge base, the model can drive computer programs [4]. In particular, this will help to achieve compliance to the standard, portability of VHDL descriptions and the kind of inter-tool communication that is required in a VHDL framework [9].

The work presented in this paper is just a first step towards using information models as a basis for language documentation and design. Future work will consider the modelling of VHDL'93 and VHDL-A. The latter project is considered as an opportunity to use information modelling in the process of language design. There is also ample opportunity for work on model verification and on model integration into practical applications.

ACKNOWLEDGEMENT

The authors wish to acknowledge the support of the Defence Research Agency (DRA), UK and the Commission of the European Communities through the ESIP (ESPRIT 8370) project. They are also grateful for the reviews of the VHDL information models by Cleland Newton of DRA, Andy Carpenter of The University of Manchester, and Serafin Olcoz and Juana Lopez of TGI, as well as other members of the VHDL community.

REFERENCES

[1] Barton D.L. Steps Toward a More Precise Relationship Between Elements of the Application Protocol Models and Formal Systems Specifications. Proceedings EII'94 Conference, The Claremont Resort, Oakland, California, 4-5 May, 1994, pp 79-88

[2] Borger E., Glasser U. and Wolfgang M. The Semantics of Behavioural VHDL'93 Descriptions. Proceedings EURO-DAC'94/EURO-VHDL'94, Grenoble, September 19-23, IEEE Press 1994, pp 500-505

[3] Standards for Electronic Design Automation, Release 1.0. CAD Framework Initiative, Inc., 4030 W.Braker Lane, Suite 550, Austin, Texas 78759 U.S.A., 1992

[4] Coyne R.D. et al. Knowledge-Based Design Systems, Addison Wesley, 1990

[5] Electronic Design Interchange Format Version 300. Electronic Industries Association / EDIF Division, 2001 Pennsylvania Avenue, N.W., Washington, DC 20006, U.S.A., EIA-618, 1993

[6] EXPRESS Language Reference Manual. ISO 10303: Part 11 Version N14, April 1991

[7] Giumale C. A. and Kahn H.J. An Information Model of Time. Proc. IEEE/ACM Design Automation Conference, Dallas 14-18 June 1993, pp 668-672

[8] Giumale C. A. An Information Model of VHDL'87 (Draft). University of Manchester, August 1994

[9] Lopez J et. al Integrating Tools in a VHDL Framework. Proceedings EII'94 Conference, The Claremont Resort, Oakland, California, 4-5 May, 1994, pp 33-41

[10] Olcoz S. and Colom J.M. The Discrete Event Simulation Semantics of VHDL. Proceedings of the International Conference of Simulation and Hardware Description Languages, Tempe, Arizona, January 1994, pp 128-134

[11] Schaefers L., Mueller W. and Wilkes W. Examination of Concepts in Existing Modelling Languages / Methodologies to Model Behavioural Semantics. Deliverable Report ECIP2/HU/008-1, 1992

[12] VHDL Language Reference Manual (IEEE Std 1076-1987/1993). IEEE, Inc., 345 East 47th Street, New York, NY 10017, USA, 1988/1993

[13] VIFASG 1076 VHDL Procedural Interface (Draft Version), VHDL Schema Definition (Draft Version). VIFASG Subgroup on Intermediate Form Definition, November 23, December 11, 1990