INTERNATIONAL STANDARD

IEC 62016

First edition
2003-12

Core model of the electronics domain

Reference number
IEC 62016:2003(E)

Publication numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series. For example, IEC 34-1 is now referred to as IEC 60034-1.

Consolidated editions

The IEC is now publishing consolidated versions of its publications. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Further information on IEC publications

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology. Information relating to this publication, including its validity, is available in the IEC Catalogue of publications (see below) in addition to new editions, amendments and corrigenda. Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is also available from the following:

- **IEC Web Site (www.iec.ch)**

- **Catalogue of IEC publications**

  The on-line catalogue on the IEC web site (http://www.iec.ch/searchpub/cur_fut.htm) enables you to search by a variety of criteria including text searches, technical committees and date of publication. On-line information is also available on recently issued publications, withdrawn and replaced publications, as well as corrigenda.

- **IEC Just Published**

  This summary of recently issued publications (http://www.iec.ch/online_news/justpub/jp_entry.htm) is also available by email. Please contact the Customer Service Centre (see below) for further information.

- **Customer Service Centre**

  If you have any questions regarding this publication or need further assistance, please contact the Customer Service Centre:

  Email: custserv@iec.ch
  Tel: +41 22 919 02 11
  Fax: +41 22 919 03 00

INTERNATIONAL STANDARD

IEC 62016

First edition
2003-12

Core model of the electronics domain

International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland
Telephone: +41 22 919 02 11 Telefax: +41 22 919 03 00 E-mail: inmail@iec.ch Web: www.iec.ch

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE XJ

For price, see current catalogue

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

CORE MODEL OF THE ELECTRONICS DOMAIN

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as “IEC Publication(s)”). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62016 has been prepared by IEC technical committee 93: Design automation.

The document was released by the feeder organization, Government Electronics and Information Technology Association, a sector of the Electronic Industries Alliance (EIA), for committee draft, comment, and review by members of IEC TC93 [1].

The text of this standard is based on the following documents:

FDIS	Report on voting
93/172/FDIS	93/176/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

[1] The EIA feeder organization retains the copyright and intellectual property of this work but provides permission for IEC to reproduce and exploit the information contained herein.

This standard does not follow the rules for the structure of International Standards given in the ISO/IEC Directives, Part 2.

The committee has decided that the contents of this publication will remain unchanged until 2012-07. At this date, the publication will be

- reconfirmed;
- withdrawn;
- replaced by a revised edition, or
- amended.

INTRODUCTION

The Core Model of the electronics domain provides a common basis for design information handled by CAD systems within the electronic domain. It is the purpose of this model to provide a conceptual representation of the electronics domain, so that the compliant CAD systems handle a similar set of concepts, thus making inter-communication, sharing and exchange of design information a much easier task. It is not the purpose of this model to describe implementation details or to provide a data representation of electronics domain information.

The Core Model of the electronics domain, Edition 1.0, is referred to as the "Core Model" throughout this document. The Core Model, in part, has been created by enhancing the industry connectivity consensus model, EDIF CFI DR Alignment Model Version 1.0 (www.edif.org).

The chosen description language for this Core Model is EXPRESS, as defined by ISO 10303-11.

It is necessary to describe the Core Model as an information model in order to provide a formal definition of the design information that shall be be recognized by the compliant CAD systems. The benefits of a formal description derive from its ability to provide an unambiguous representation of concepts, attributes and relationships, and the global rules and constraints that may be applied. By having such a description, it is possible to check the consistency and the correctness of the model as well as to provide a reliable starting-point for further development. It also facilitates the design of correct electronics CAD implementations based on this Core Model, as the actual implementation methods can be checked against the model.

This Core Model includes connectivity, hierarchy and design information for the electronics domain. Future parts of this Core Model standard may be extended to include other categories of information (for example, *cell_representation*, schematic representation, the PCB domain, symbols and display information).

In order to facilitate the creation of other parts of this Core Model standard, some objects have been used in this Core Model to facilitate support for other parts of the electronics domain. There are two types of such objects.

- Entities, such as *cell_representation*, are important concepts that provide support for defining other Core Model parts.
- Constraints: Some of constraints of this Core Model use conditions that are always true. They have been written in this way in order to ensure that they remain valid when the model is extended.

CORE MODEL OF THE ELECTRONICS DOMAIN

1 Scope and object

This International Standard provides the semantics definitions for the following categories of information related to electronic circuit designs. Each category of design information is modelled as an EXPRESS schema.

The Core Model consists of 10 schemas. Each of them is presented in this document as a separate chapter. At the beginning of each chapter, a description of the corresponding schema is provided.

- The *hierarchy_model* schema describes the hierarchical information of a cell, i.e. the way a cell may be divided into other cells.

  A circuit may be divided into cells which, in turn, may be further subdivided into other cells, thus creating a hierarchy. The hierarchy information describes the cells, the possible cell representations and their instances.
- The *design_hierarchy_model* schema describes the annotation on an occurrence hierarchy.

  The definition of a design requires that specific representations (views) of design objects in the hierarchy are selected. This unambiguously creates a configured design hierarchy. This concept is similar to the configuration of a design in VHDL and is related to view selection mechanisms of other electronics design domain information models in industry. The design hierarchy identifies top-level design cells and may provide annotated design-specific data into the elaborated hierarchy.
- The *connectivity_view_model* schema describes the connectivity information of a cell.

  This describes the way in which the circuits are connected in order that information or energy may flow from one part of a design or product to another. The Core Model subdivides this information into
  - the *logical_connectivity_model* schema which describes the connectivity for a given level of a hierarchy.

    Logical connectivity information describes the bit level, abstract electrical connectivity for a given level of a hierarchy, in terms of signals and signal groups.
  - the connectivity_structure_model schema which describes the structural connectivity of a connectivity view.

    Structural connectivity information describes the connectivity, for a given level of a hierarchy, from the structural point of view. The structural connectivity is specified in terms of busses, nets and rippers. Such a structural representation is used to provide support for physical implementation and annotation.
- The *library_model* schema describes the technology information contained in a library, as well as the reusable objects and data.

  A library provides a means of grouping cell definitions. A library may be used to group other classes of reusable objects and information as well. Information in a library may be related in terms of technology information.
- The *information_base_model* schema describes the information in an information base.

  The information_base describes the kind of information that can be found directly in an information base.

In addition, the following information is also included in the model.

- The *design_management_model* schema provides the design management information.

  This information is needed to trace back to the source or the owner of the data.
- The *documentation_model* schema describes the documentation provided for an object.
- The *support_definition_model* schema contains the definition of some auxiliary entities, types and functions that are used by several schemas.

Names of objects used in this Core Model standard were chosen to be the same as the names of the similar objects and concepts in existing electronics domain information models wherever possible.

2 Reference documents

IEC 61690-1:2000, *Electronic design interchange format (EDIF) – Part 1: Version 300*

IEC 61690-2:2000, *Electronic design interchange format (EDIF) – Part 2: Version 400*

ISO 13030-11:1994, *Industrial automation systems and integration – Product representation and exchange – Part 11: Description methods: The EXPRESS-I language reference manual*

3 General modelling issues

This standard describes the general modelling techniques and conventions used by the Core Model. The most important concept discussed here is that of relationship between entities (ownership and reference). In addition, issues such as uniqueness in aggregates, default values, empty sets and model topology are presented.

3.1 Ownership and reference

An object is said to be the “owner” of another object, if the latter can only exist in the context of the former. From the point of view of the information model, this means that an instance of the “owned” object can exist if, and only if, there is an instance of the owner object. Any given instance of an object must have exactly one owner. Some examples of owner relationships are provided below.

By contrast, if an object references another object, the existence of the latter is not dependent on the existence of the former. In the information model this means that the existence of an instance of the referenced object is not related to the existence of any instance of the object that references it. A given instance may be referenced by any number of other instances unless stated otherwise by a constraint.

Generally speaking, the referencing mechanism provides a way of sharing data whereas the ownership dependence is used to create a scope for the objects and to control their existence.

The difference between the two types of relationship is important because of their effects in an actual implementation. If the implementation provides a static representation of data (such as an EDIF file), an owned object can be textually contained in the definition of the owner, whereas a referenced object is used by the referencing object but may be declared in another context. If the system allows dynamic representation of the data (like CFI DR PI), the difference between ownership and reference is reflected in the process of object creation and destruction. Indeed, whenever an object is destroyed, its owned objects are destroyed too, unlike the referenced objects which continue to exist.

The EXPRESS language does not provide a direct method of specifying whether an object is owned or referenced. However, the Alignment Model represents the two types of relationship in different ways by using the techniques and conventions described below.

3.1.1 Ownership

Let us consider the example presented in Figure 1 which shows the owner relationship between a *library* and its contained *cells*:

```
ENTITY library
  cells : OPTIONAL SET [1:?] OF cell;
  ...
END_ENTITY;


ENTITY cell
  ...
INVERSE
  containing_library : library FOR cells;
END_ENTITY;
```

Figure 1 – The owner relationship

In this example, ownership is modelled straightforwardly by using the EXPRESS INVERSE clause. Its meaning is that, for every instance of a *cell,* there is exactly one instance of the *library* that contains that *cell* instance.

There are cases, however, when an object may have several potential owners. The example in Figure 2 shows that a *documentation* object may be created in the context of a *connectivity_generic_net* or a *connectivity_generic_bus*, etc.

The owner relationship is modelled by using the INVERSE clause. Its meaning is that there may exist, at most, one instance of the *connectivity_generic_net* and, at most, one instance of a *connectivity_generic_bus*, etc, that contain a given *documentation* instance. However, any instance of the *documentation* may exist as a member of the "document" SET in a *connectivity_generic_net* or as a member of the "document" SET in a *connectivity_generic_bus*, etc, but not as all of them. Therefore, a domain rule (the WHERE clause) is used in order to ensure that there is only one owner of the *documentation* instance in the database. Of course, there are other objects that reference that *documentation* instance.

```
ENTITY connectivity_generic_net
  ABSTRACT SUPERTYPE OF (ONEOF(connectivity_net,
                               connectivity_sub_net));
  ...
  document    : OPTIONAL SET [1:?] OF documentation;
  ...
END_ENTITY;


ENTITY connectivity_generic_bus
  ABSTRACT SUPERTYPE OF (ONEOF(connectivity_bus,
                               connectivity_bus_slice,
                               connectivity_sub_bus));
  ...
  document    : OPTIONAL SET [1:?] OF documentation;
  ...
END_ENTITY;


ENTITY documentation;
  ...
INVERSE
  containing_generic_net:
    SET [0:1] OF connectivity_generic_net FOR document;
  containing_generic_bus:
    SET [0:1] OF connectivity_generic_bus FOR document;
  ...
WHERE
  containment_constraint:
    (* Each "documentation" is defined in only one place. *)
    SIZEOF(containing_generic_net) +
    SIZEOF(containing_generic_bus) +
  ...
END_ENTITY;
```

Figure 2 – Owner relationship with multiple potential owners

3.1.2 Reference

The reference mechanism can be used to share common data between several objects. Let us consider a *master_logical_port* which has a default connection to a *global_port*. Since it is possible for several *master_logical_ports* to have the same default connection, the *global_port* is referenced by the *master_logical_port*. In this case, the *global_port* does not contain an INVERSE attribute for the *master_logical_port* because it is not owned by it. In order to increase the readability of the model, the references to other objects are tagged as such. Figure 3 gives an example of object referencing.

```
ENTITY master_logical_port;
  default_connection : OPTIONAL global_port; -- reference
  ...
END_ENTITY;
```

Figure 3 – The reference mechanism

3.2 Uniqueness by value

EXPRESS defines the uniqueness in aggregates to be "by reference". This means that a set cannot contain the same object twice but it may contain two different objects with exactly the same value. Therefore, a special EXPRESS function is used whenever uniqueness by value is required, as shown in Figure 4.

```
ENTITY library
  document            : OPTIONAL SET [1:?] OF documentation;
  status_of_copyright : OPTIONAL SET [1:?] OF copyright;
  status_of_written   : OPTIONAL SET [1:?] OF written;
  ...
WHERE
  unique_status_of_copyright:
    value_unique(status_of_copyright);
  unique_status_of_written:
    value_unique(status_of_written);
  unique_document:
    value_unique(document);
END_ENTITY;
```

Figure 4 – Uniqueness by value

The example shows that a *library* should not contain two identical documentation, copyright information or author information, i.e. it cannot contain two instances of *documentation*, *copyright* or *written* with the same value.

3.3 Default values

The default values, which are used by the CAD systems, are relevant to the actual implementations rather than to the information models. Therefore, the Core Model does not contain a specification of these values.

3.4 Optional versus empty sets

In EXPRESS, a set which may have no elements can be described by either

- `OPTIONAL SET [1:upper_limit] OF base_type`, or by
- `SET [0:upper_limit] OF base_type`.

However, the concepts modelled by the two descriptions are quite different. An `OPTIONAL SET` shows the fact that the set may or may not exist whereas a `SET[0 : upper_limit]` shows that the set always exists but it may be empty. The Core Model uses the former method.

3.5 Model topology

The topology is the graph representation of the relationships between the concepts defined in the model. Two possible approaches have been considered.

- A tree representation in which there is a concept (the root of the tree) that is a generalization of all the other concepts defined in the model. An example of Information Model that uses this topology is the CFI DR IM.
- A forest representation in which there are several categories of concepts that do not have any common features. An example of Information Model that uses this topology is the EDIF IM.

The forest representation has been chosen for the Core Model. An advantage of this representation (as opposed to the tree representation) is that it reduces the maximum depth of the leaves in the model hierarchy.

4 Concepts

This chapter describes the fundamental concepts defined by the Core Model. These concepts may be thought of as creating a hierarchy which is abstract at its higher level and becomes progressively more detailed. At the highest level of the structure, an *information_base* may contain *designs* and *libraries*. Each *library* is a collection of *cells* which are grouped according to a set of common characteristics. A *cell* is the main design object which may be instantiated later in another *cell*, thus creating a design hierarchy. A *cell* may be connected to other *cells* by means of its interface. These concepts are described in the following sections.

4.1 The information base

An *information_base* describes the data that can be found in the database of a compliant CAD system. This information may include *libraries*, *designs*, *global_ports* and *global_port_bundles* and *unit* definitions. The Core Model is an information model that describes one information base. All the entities defined in the Core Model belong, directly or indirectly, to the *information_base* object. The model of the *information_base* entity can be found in the *information_base_model* schema. A partial EXPRESS-G diagram is also available.

4.2 Global ports and global port bundles

Certain ports such as GND, VCC or clocks are used in common by several modules of an electronic circuit. The Core Model describes this by using the concept of global port, which is modelled by the *global_port* entity. The *global_ports* are defined in the *information_base* and are, therefore, visible in all the *cells* and *designs* defined in this information base.

It is sometimes convenient to address several ports as if they were a single port. In order to do this the Core Model uses the concept of port bundle. A port bundle provides a means of creating a structured port by grouping other ports and/or port bundles defined in the same context. In the case of *global_ports*, the port bundle is modelled by the *global_port_bundle* entity.

Both the *global_port* and the *global_port_bundle* are described in the *information_base_model* schema. A partial EXPRESS-G diagram is also available.

4.3 Libraries

A *library* provides a means of grouping design units (*cell* definitions). In future versions of the Core Model a *library* will contain other classes of reusable objects, as well.

A *library* may be locally defined, in which case the definition of the contained *cells* is included in the information base. *Libraries* may also be externally defined, which implies that the details of the *cell* definitions are not available within the information base.

The *library_model* schema describes the model for the *library* entity. A partial EXPRESS-G diagram is also available. According to where the contained data can be found, the library entity is subtyped into *internal_library* and *external_library*. The *internal_library* corresponds to the case where the library is locally defined. Therefore, since all the data is present in the *information_base*, an *internal_library* may contain implementation information. An *external_library* corresponds to the case where the library data is not contained in the *information_base* and, therefore, it does not include implementation details.

4.4 Cells

A *cell* is the basic design unit described by the Core Model. In order to be able to represent complex circuits, a *cell* can be divided into other *cells*, thus creating a hierarchy. The model of the *cell* entity is given in the *hierarchy_model* schema. A partial EXPRESS-G *cell* entity is subtyped into *internal_cell* and *external_cell* according to whether it describes information defined locally or not. An *internal_cell* contains locally defined data and may, therefore, contain implementation information. An *external_cell* does not include implementation details. Since an *internal_cell* contains data defined in the current *information_base*, it can only belong to an *internal_library*. Similarly, an *external_cell* belongs to an *external_library*.

4.5 Clusters and cell representation sets

A *cell* is described by means of *cell_representations*. *Cell_representations* that share the same interface can be grouped into *clusters*. A *cell* can contain more than one *cluster*. A *cluster* may be instantiated in another cell view, thus creating a design hierarchy. A *cluster* is subtyped into *internal_cluster* and *external_cluster* which are used by *internal_cell* and *external_cell* respectively. An *internal_cluster* may only contain *internal_cell_representations*, and an *external_cluster* only *external_cell_representations*. The model of the *cluster* entity is given in the *hierarchy_model* schema. A partial EXPRESS-G diagram is also available.

Cell representation sets are used to group *cell_representations* of the same *cell* that have a close relationship based on reasons other than a common interface. Simple annotation is provided within the *cell representation set* in order to specify the reason for the grouping. The model for the *cell_representation_set* entity can be found in the *hierarchy_model* schema.

4.6 Cell representations

A *cell_representation* describes chosen aspects of a *cell*. At present, the Core Model supports only one type of *cell_representation*, the connectivity view. Future versions of the model will be extended, however, with other view types such as schematic and pcblayout, as well as with symbols. A *cell_representation* may be either an *internal_cell_representation* or an *external_cell_representation*. An *internal_cell_representation* belongs to an *internal_cluster* and describes an *internal_cell*. Similarly, an *external_cell_representation* belongs to an *external_cluster* and describes an *external_cell*. An *internal_cell_representation* can establish a direct relationship with other *cell_representations* in order to provide versioning and data management control information. Hence, an *internal_cell_representation* may be a new version of another *cell_representation* of the same type or may be derived from another *cell_representation*. If the *internal_cell_representation* is derived from another *cell_representation*, their type can be different. The interface of a *cell_representation* is obtained from its containing *cluster*.

The model of the *cell_representation* entity can be found in the *hierarchy_model* schema. A partial EXPRESS-G *internal_connectivity_view* and *external_connectivity_view* are given in the *connectivity_view_model* schema.

4.7 Master ports and master port bundles

The interface of a *cluster* consists of ports and port bundles. These objects represent the only places where a connection can be made to a *cell*. The interface of a *cluster* is described by the *cluster_interface* entity.

In the Core Model, each port in the cluster interface is described by a *master_logical_port* entity which describes its logical connectivity information. A *master_logical_port* can be associated with one or several *master_structure_ports* which describe the structural connectivity information of the port. It is possible that a *master_logical_port* is not associated with any *master_structure_port*.

The size of the *master_logical_port* and *master_structure_port* is always one. The *master_logical_port* entity is subtyped into *input_master_logical_port*, *output_master_logical_port*, *bidirectional_master_logical_port* and *unspecified_direction_master_logical_port*, which correspond to the possible directions of the port. In addition, a port contains information such as its designator, its properties and its external load capacitance. It is possible to specify a *global_port* as the default connection for a *master_logical_port*.

It is sometimes convenient to address several ports as if they were a single port. In order to be able to do this, a grouping method is provided. The Core Model uses the concept of port bundle in order to describe a structured port. A port bundle in the cluster interface is described by a *master_logical_port_bundle* entity, which groups *master_logical_ports* and/or other *master_logical_port_bundles* defined in the same interface. The same *master_logical_port* may be referenced by more than one *master_logical_port_bundle*. However, it cannot be referenced more than once by the same *master_logical_port_bundle*, either directly or indirectly.

The availability of a *master_logical_port_bundle* for structural connectivity is indicated by the *master_structure_port_bundle* entity. A *master_logical_port_bundle* can be associated with more than one *master_structure_port_bundle*.

A structured port can be defined in the interface of a *cluster* by using the *port_structure* entity. A *port_structure* describes a possible structuring of the structure ports and port bundles of a cluster interface. It can represent either an ordered or an unordered structured port.

The models for the *master_logical_port*, *master_structure_port*, *master_logical_port_bundle*, *master_structure_port_bundle* and *port_structure* entities are given in the *hierarchy_model* schema. Partial EXPRESS-G diagrams for *cluster_interface* and *port_structure* are also available.

4.8 Instances

A *cluster* may be instantiated in a cell view, thus enabling the creation of a design hierarchy. The *instance* entity models the instance of a *cluster*. The interface of the *instance* consists of *instance_structure_ports* and *instance_structure_port_bundles* which must reference *master_structure_ports* and *master_structure_port_bundles* defined in the interface of the instantiated *cluster*. If the width of the *instance* is greater than one, the *instance* represents an arrayed *instance*.

The advantage of the instantiation mechanism is that the choice of a particular *cell_representation* can be delayed until the complete *design* is configured. For example, a connectivity view may contain *instances* of *clusters* of other *cells*. When the connectivity view is configured, *instances* are configured by selecting suitable views.

An *internal_connectivity_view* can define, for each of its instances, a *connectivity_instance_implementation* which describes the structural information associated with an *instance*. Since the only type of *cell_representation* defined in the current model is the connectivity view, the only significant structural information is the port structuring.

The model for the *instance* entity can be found in the *hierarchy_model* schema. A partial EXPRESS-G diagram is also available.

4.9 Instance ports and instance port bundles

The interface of the *instance* consists of *instance_structure_ports* and *instance_structure_port_bundles*. The *instance_structure_port* is defined by either a *connectivity_generic_bus* or by a *connectivity_generic_net* and associates a *master_structure_port* with an *instance*. The information contained in the *master_logical_port* associated with the referenced *master_structure_port* may be overridden by the *instance* if it defines an *instance_port_attributes* entity associated with the *master_logical_port*.

The *instance_port_attributes* entity is subtyped into *input_instance_port_attributes*, *output_instance_port_attributes*, *bidirectional_instance_port_attributes* and *unspecified_ direction_instance_port_attributes*, corresponding to the possible types of the associated *master_logical_port*.

An *instance_structure_port_bundle* entity references a *master_structure_port_bundle* which must be defined in the interface of the instantiated *cluster*. The *instance* may override some of the information contained in the *master_logical_port_bundle* associated with the referenced *master_structure_port_bundle* by defining an *instance_port_bundle_attributes* entity.

The Core Model defines the *instance* entity to be an indexed collection of *cluster* instances. The number of elements in the collection is given by the "width" attribute in the *instance* entity. A port in the interface of a member of the *instance* is modelled by the *instance_member_ logical_port* and *instance_member_structure_port* entities.

The models for the *instance_structure_port*, *instance_member_logical_port*, *instance_ member_structure_port*, *instance_structure_port_bundle*, *instance_port_attributes* and *instance_port_bundle_attributes* entities are given in the *hierarchy_model* schema.

5 Connectivity

The only view type supported by the model at present is the connectivity view. Future versions of the model may contain other views that support connectivity definitions such as schematic and pcblayout. The Core Model differentiates between two levels of connectivity information within the connectivity view.

- Logical connectivity information

  The logical connectivity describes the bit level, abstract electrical connectivity for a given level of a hierarchy, in terms of signals and signal bundles. Every type of view that supports connectivity definitions contains similar connectivity information.

- Structural connectivity information

  The structural connectivity describes the connectivity, for a given level of a hierarchy, from the structural point of view. The structural connectivity is specified in terms of buses, nets and rippers. Such a structural representation is used to provide support for physical implementation and annotation. The structural connectivity information is view-type specific. Indeed, the manner in which the connectivity view is structured is unlikely to be the same as that of a future schematic or pcb view.

5.1 Logical connectivity

The model for the logical connectivity can be found in the *logical_connectivity_model* schema. A partial EXPRESS-G diagram is also available. The logical connectivity of a view is described in terms of *signals* and *signal_bundles*.

5.1.1 Signals

A *signal* is defined as an object that provides a means by which all the *global_ports*, *instance_member_logical_ports* and *master_logical_ports* may be logically electrically common at a a given level of hierarchy, and to enable the logical flowing, sharing and exchanging of information and energy between the applicable blocks.

A *signal* is always one bit wide and is defined within an *internal_connectivity_view*. All *global_ports* joined by a *signal* are defined in the containing *information_base*. All *instance_member_logical_ports* joined by a *signal* must reference *instances* in the containing view. All *master_logical_ports* joined by a *signal* are defined in the interface of the containing *cluster*.

5.1.2 Signal_bundle

A *signal_bundle* provides a grouping mechanism for *signals*. A *signal_bundle* is defined within a view and is used to support structural connectivity. However, it does not provide additional connectivity information. A *signal_bundle* is specified as an ordered list of one or more *signals* or *signal_bundles* in the same containing view. The structure of a *signal_bundle* is independent of port ordering or port structure. No two *signal_bundles* may have the same structure of members. The same *signal* may be referenced by more than one *signal_bundle* and may be referenced more than once in the same *signal_bundle*. Since a *signal_bundle* may contain *signals* and *signal_bundles*, it creates a signal hierarchy as shown in the example in Figure 5.

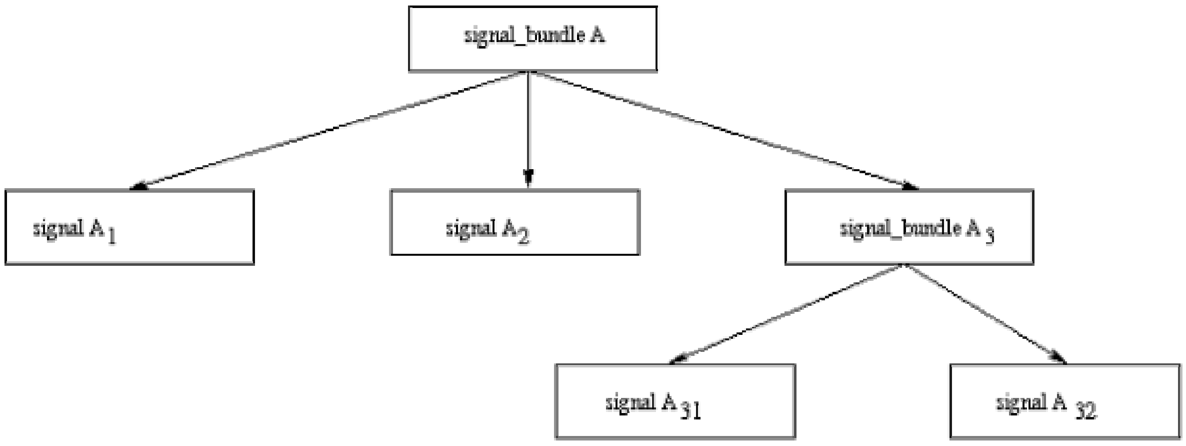

Figure 5 – An example of signal hierarchy

Such a hierarchical structure is used to support the structural connectivity expressed by buses.

5.2 Structural connectivity with wide instances

When an interconnect (net or bus) joins a port or a port bundle on a wide *instance*, the pattern of connectivity will be either in the commoned style or in the fanned-out style. For the commoned style, the joined *instance_structure_ports* or *instance_structure_port_bundles* reference *master_structure_ports* or *master_structure_port_bundles* whose size is the same as that of the interconnect. For the fanned-out style, the joined *instance_structure_ports* are flattened to lists of *instance_member_structure_ports*. The length of these lists is equal to the size of the interconnect. In the case of *instance_structure_port_bundles*, their size is equal to the size of the *master_logical_port_bundle* associated with the *master_structure_port_bundle* referenced by the *instance_structure_port_bundle*. The size of the *instance_structure_port_bundle* must be equal to the size of the joined interconnect (whose size is equal to the size of its associated *signal* or *signal_bundle*). A net is associated with a *signal* and, therefore, its size is always one. A bus is associated with a *signal_bundle* and, therefore, its size is equal to the length of the list created by flattening the *signal_bundle*. Several examples are given below.

5.2.1 Joining ports on instances in the commoned style

Figure 6 presents an example of a net joining a port on a wide *instance*, using the commoned style. The wide *instance* "I" has a width of three. The instantiated *cluster* defines a *master_structure_port* 'P' in the *cluster_interface*.