IEC 62014-4

Edition 1.0 2015-03

INTERNATIONAL STANDARD

IEEE Std 1685™-2009

IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040

ISBN 978-2-8322-2265-2

Contents

IP-XACT, STANDARD STRUCTURE FOR PACKAGING, INTEGRATING, AND REUSING IP WITHIN TOOL FLOWS

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.

   IEEE Standards documents are developed within IEEE Societies and Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. IEEE develops its standards through a consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of IEEE and serve without compensation. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards. Use of IEEE Standards documents is wholly voluntary. IEEE documents are made available for use subject to important notices and legal disclaimers (see http://standards.ieee.org/IPR/disclaimers.html for more information).

   IEC collaborates closely with IEEE in accordance with conditions determined by agreement between the two organizations.

2) The formal decisions 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. The formal decisions of IEEE on technical matters, once consensus within IEEE Societies and Standards Coordinating Committees has been reached, is determined by a balanced ballot of materially interested parties who indicate interest in reviewing the proposed standard. Final approval of the IEEE standards document is given by the IEEE Standards Association (IEEE-SA) Standards Board.

3) IEC/IEEE Publications have the form of recommendations for international use and are accepted by IEC National Committees/IEEE Societies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC/IEEE Publications is accurate, IEC or IEEE 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 (including IEC/IEEE Publications) transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC/IEEE Publication and the corresponding national or regional publication shall be clearly indicated in the latter.

5) IEC and IEEE do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC and IEEE are not responsible for any services carried out by independent certification bodies.

6) All users should ensure that they have the latest edition of this publication.

7) No liability shall attach to IEC or IEEE or their directors, employees, servants or agents including individual experts and members of technical committees and IEC National Committees, or volunteers of IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board, 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/IEEE Publication or any other IEC or IEEE 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 implementation of this IEC/IEEE Publication may require use of material covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. IEC or IEEE shall not be held responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patent Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility.

International Standard IEC 62014-4/ IEEE Std 1685-2009 has been processed through IEC technical committee 91: Electronics assembly technology, under the IEC/IEEE Dual Logo Agreement.

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

IEEE Std	FDIS	Report on voting
1685 (2009)	91/1207/FDIS	91/1226/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.

The IEC Technical Committee and IEEE Technical Committee have decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

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

IEEE Std 1685™-2009

IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows

Sponsor
Design Automation Standards Committee
of the
IEEE Computer Society
and the
IEEE Standards Association Corporate Advisory Group

Approved 9 December 2009
IEEE SA-Standards Board

Grateful acknowledgment is made to The SPIRIT Consortium, Inc., for permission to use the following source material:

IP-XACT 1.2 and IP-XACT 1.5

Abstract: Conformance checks for eXtensible Markup Language (XML) data designed to describe electronic systems are formulated by this standard. The meta-data forms that are standardized include: components, systems, bus interfaces and connections, abstractions of those buses, and details of the components including address maps, register and field descriptions, and file set descriptions for use in automating design, verification, documentation, and use flows for electronic systems. A set of XML schemas of the form described by the World Wide Web Consortium (W3C®) and a set of semantic consistency rules (SCRs) are included. A generator interface that is portable across tool environments is provided. The specified combination of methodology-independent meta-data and the tool-independent mechanism for accessing that data provides for portability of design data, design methodologies, and environment implementations.
Keywords: abstraction definitions, address space specification, bus definitions, design environment, EDA, electronic design automation, electronic system level, ESL, implementation constraints, IP-XACT, register transfer level, RTL, SCRs, semantic consistency rules, TGI, tight generator interface, tool and data interoperability, use models, XML design meta-data, XML schema

IEEE Introduction

This introduction is not part of IEEE Std 1685-2009, IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows.

The purpose of this standard is to provide the electronic design automation (EDA), semiconductor, electronic design intellectual property (IP) provider, and system design communities with a well-defined and unified specification for the meta-data that represents the components and designs within an electronic system. The goal of this specification is to enable delivery of compatible IP descriptions from multiple IP vendors; better enable importing and exporting complex IP bundles to, from, and between EDA tools for system on chip (SoC) design environments (DEs); better express configurable IP by using IP meta-data; and better enable provision of EDA vendor-neutral IP creation and configuration scripts (*generators*). The data and data access specification is designed to coexist and enhance the hardware description languages (HDLs) presently used by designers while providing capabilities lacking in those languages.

The SPIRIT Consortium is a consortium of electronic system, IP provider, semiconductor, and EDA companies. IP-XACT enables a productivity boost in design, transfer, validation, documentation, and use of electronic IP and covers components, designs, interfaces, and details thereof. The data specified by IP-XACT is extensible in locations specified in the schema.

IP-XACT enables the use of a unified structure for the meta specification of a design, components, interfaces, documentation, and interconnection of components. This structure can be used as the basis of both manual and automatic methodologies. IP-XACT specifies the tight generator interface (TGI) for access to the data in a vendor-independent manner.

This standardization project provides electronic design engineers with a well-defined standard that meets their requirements in structured design and validation, and enables a step function increase in their productivity. This standardization project will also provide the EDA industry with a standard to which they can adhere and that they can support in order to deliver their solutions in this area.

The SPIRIT Consortium has prepared a set of bus and abstraction definitions for several common buses. It is expected, over time, that those standards groups and manufacturers who define buses will include IP-XACT eXtensible Markup Language (XML) bus and abstraction definitions in their set of deliverables. Until that time, and to cover existing useful buses, a set of bus and abstraction definitions for common buses has been created.

A set of reference bus and abstraction definitions allows many vendors who define IP using these buses to easily interconnect IP together. The SPIRIT Consortium posts these for use by its members, with no warranty of suitability, but in the hope that these will be useful. The SPIRIT Consortium will, from time-to-time, update these files and if a Standards body wishes to take over the work of definition, will transfer that work to that body.

These reference bus and abstraction definition templates (with comments and examples) are available from the public area of The SPIRIT Consortium Web site.[a]

[a]Available at http://www.spiritconsortium.org.

Notice to users

Laws and regulations

Users of these documents should consult all applicable laws and regulations. Compliance with the provisions of this standard does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

This document is copyrighted by the IEEE. It is made available for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making this document available for use and adoption by public authorities and private users, the IEEE does not waive any rights in copyright to this document.

Updating of IEEE documents

Users of IEEE standards should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Standards Association website at http://ieeexplore.ieee.org/xpl/standards.jsp, or contact the IEEE at the address listed previously.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA website at http://standards.ieee.org.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/index.html.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents or patent applications for which a license may be required to implement an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows

1. Overview

This clause explains the scope and purpose of this standard; gives an overview of the basic concepts, major semantic components, and conventions used in this standard; and summarizes its contents.

1.1 Scope

This standard describes an eXtensible Markup Language (XML) schema[1] for meta-data documenting *intellectual property* (IP) used in the development, implementation, and verification of electronic systems and an *application programming interface* (API) to provide tool access to the meta-data. This schema provides a standard method to document IP that is compatible with automated integration techniques.The API provides a standard method for linking tools into a *system development* framework, enabling a more flexible, optimized development environment. Tools compliant with this standard will be able to interpret, configure, integrate, and manipulate IP blocks that comply with the IP meta-data description. The standard is based on version 1.4 IP-XACT of The SPIRIT Consortium. The standard is independent of any specific design processes. It does not cover those behavioral characteristics of the IP that are not relevant to integration.

[1]Information on references can be found in Clause 2.

1.2 Purpose

This standard enables the creation and exchange of IP in a highly automated design environment.

1.3 Design environment

The IP-XACT specification is a mechanism to express and exchange information about design IP and its required configuration.[2] While the IP-XACT description formats are the core of this standard, describing the IP-XACT specification in the context of its basic use model, the design environment (DE), more readily depicts the extent and limitations of the semantic intent of the data. The DE coordinates a set of tools and IP, or expressions of that IP (e.g., models), through the creation and maintenance of meta-data descriptions of the system on chip (SoC) such that its system design and implementation flows are efficiently enabled and reuse centric.

The use of the IP-XACT specified formats and interfaces are shown, in **bold**, in Figure 1 and described in the following subclauses.

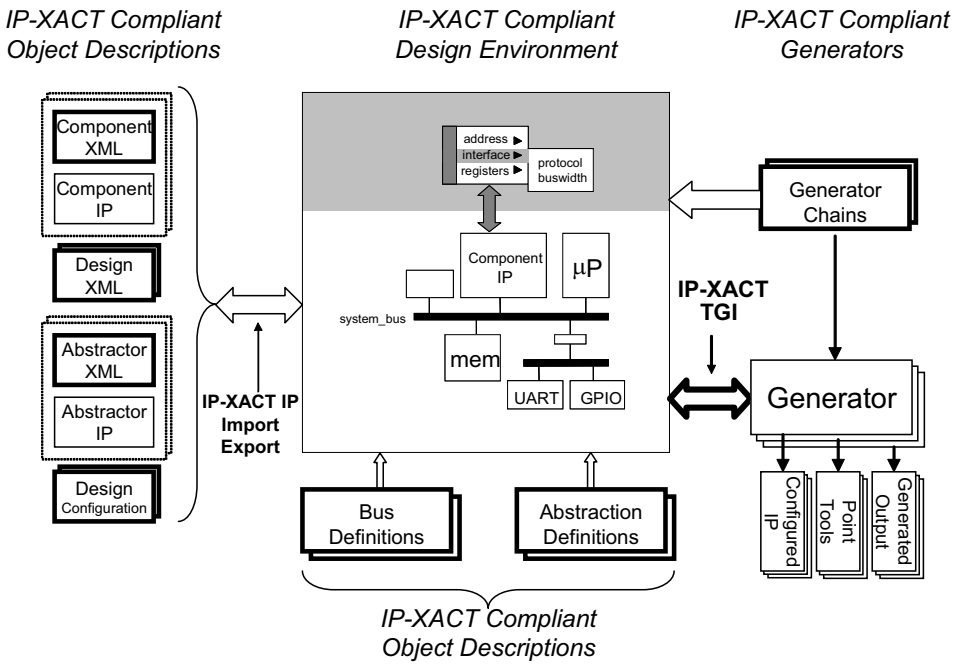

Figure 1—IP-XACT design environment

1.3.1 IP-XACT design environment

A DE enables the designer to work with IP-XACT design IP through a coordinated front-end and IP design database. These tools create and manage the top-level meta-description of system design and may provide two basic types of services: *design capture*, which is the expression of design configuration by the IP

[2]IP-XACT uses the World Wide Web Consortium (W3C®) standard for the XML version 1.0 data (http://www.w3.org/TR/2000/REC-xml-20001006). The valid format of that XML data is described in a *schema* by using the Schema Description Language described therein. W3C is a registered trademark of the World Wide Web Consortium.

provider and design intent by the IP user; and *design build*, which is the creation of a design (or design model) to those intentions.

As part of design capture, a system design tool shall recognize the structure and configuration options of imported IP. In the case of *structure*, this implies both the structure of the design [e.g., how specific pin-outs refer to lines in the hardware description language (HDL) code] as well as the structure of the IP package (e.g., where design descriptions and related generators are provided in the packaged IP data-structure). In the case of *configuration*, this is the set of options for handling the imported IP (e.g., setting the base address and offset, bus width) that may be expressed as configurable parameters in the IP-XACT meta-data.

As part of design build, generators may be provided internally by a system design tool to achieve the required IP integration or configuration, or provided externally (e.g., by an IP provider) and launched by the system design tool as appropriate.

The *system design tool set* defines a DE where the support for conceptual context and management of IP-XACT meta-data resides. However, the IP-XACT specifications make no requirements upon system design tool architecture or a tool's internal data structures. To be considered IP-XACT v1.5 enabled, a system design tool shall support the import/export of IP expressed with valid IP-XACT v1.5 meta-data for both component IP and designs, and it needs to support the tight generator interface (TGI) for interfacing with external generators (to the DE).

1.3.2 IP-XACT object descriptions

The IP-XACT schema is the core of the IP-XACT specification. There are seven top-level schema definitions. Each schema definition can be used to create object descriptions of the corresponding type.

- A *bus definition* description defines the type attributes of an bus.
- An *abstraction definition* description defines the representation attributes of a bus.
- A *component* description defines an IP or interconnect structure.
- A *design* description defines the configuration of and interconnection between components.
- An *abstractor* description defines an adaptor between interfaces of two different abstractions.
- A *generator chain* description defines the grouping and ordering of generators.
- A *design configuration* description defines additional configuration information for a generator chain or design description.

1.3.3 Object interactions

An object description contains a unique identifier in the header. The identifier in IP-XACT terms is called a VLNV after the four elements that define its value: vendor, library, name, and version. See C.6 for further details on a VLNV. This VLNV is used to create a reference from one description to another. The links between these objects are illustrated in Figure 2. The arrows (A → B) illustrate a reference of one object to another (e.g., reference of object `B` from object `A`).

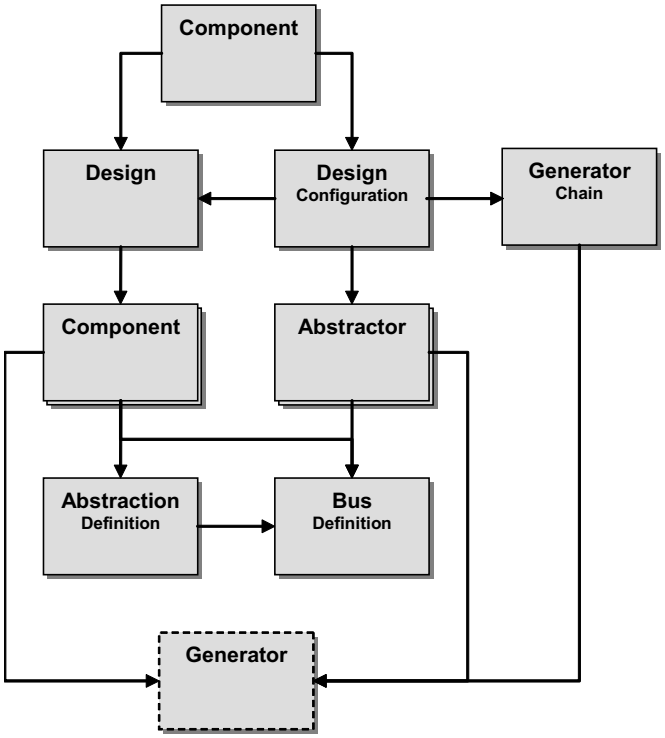

Figure 2—IP-XACT object interactions

1.3.4 IP-XACT generators

Generators are executable objects (e.g., scripts or binary programs) that may be integrated within a DE (referred to as *internal*) or provided separately as an executable (referred to as *external*). Generators may be provided as part of an IP package (e.g., for configurable IP, such as a bus-matrix generator) or as a way of wrapping point tools for interaction with a DE (e.g., an external design netlister, external design checker).

An internal generator may perform a wide variety of tasks and may access IP-XACT compliant meta-data by any method a DE supports. IP-XACT does not describe these protocols.

An *external generator* (often referred to as a *TGI generator*) is an executable program or script invoked from within a DE to query or configure design descriptions and their related component and abstractor descriptions. External generators can use the TGI to *access* their IP-XACT meta-data descriptions (as currently loaded into the DE) and perform the various operations associated with those descriptions. In addition, external generators shall only *operate* upon IP-XACT compliant meta-data through the defined TGI, see 1.3.6.

Generators can be referenced from a component, abstractor, or generator chain description. Generators can also be grouped and ordered in generator chain descriptions and those chain descriptions contained inside other chain descriptions. This sequencing of generators is *critical* for providing script-based support for SoC flow creation.

1.3.5 IP-XACT design environment interfaces

There are two obvious interfaces expressed in Figure 1: from the DE to the external IP libraries and from the DE to the generators. In the former case, the IP-XACT specifications are *neutral* regarding the design tool

interfaces to IP repositories. Being able to read and write IP with IP-XACT meta-data is required; however, the *formal interaction* between an external IP repository and a DE is not specified.

1.3.6 Tight generator interface

The *tight generator interface* (TGI) is the method a generator uses to efficiently access a design or component description in a DE-independent and generator-language-independent manner. Therefore, a generator running on two different DEs produces the same results. The DE and the generator communicate with each other by sending messages utilizing the Simple Object Access Protocol (SOAP) standard version 1.2[3] specified in the Web Services Description Language (WSDL) version 1.1.[4] SOAP provides a simple means for sending XML-format messages using the Hypertext Transfer Protocol (HTTP) or other transport protocols. IP-XACT supports using an HTTP protocol or a file protocol.

The SOAP messages passed between the generator and the DE allow the generator to get all information about the design interconnections (which contain components and abstractors), provide set information for any configurable elements in a component or abstractor, and make simple modifications of the design description. For additional details on the DE generator invocation and the SOAP messages passed between the generator and the DE, see Annex G.

1.3.7 Design intellectual property

IP-XACT is structured around the concept of IP reuse. *Electronic design intellectual property*, or IP, is a term used in the ED community to refer to a reusable collection of design specifications that represent the behavior, properties, and/or representation of the design in various media. The name IP is partially derived from the common practice of considering a collection of this type to be the intellectual property of one party. Both hardware and software collections are encompassed by this term.

These collections may include the following:

a) Design objects—This can include the following:
   1) Transaction-level modeling (TLM) descriptions: SystemC® and SystemVerilog [5]
   2) Fixed HDL descriptions: Verilog®, VHDL[6]
   3) Configurable HDL descriptions (e.g., bus-fabric generators)
   4) Design models for register transfer level (RTL) and transactional simulation (e.g., compiled core models)
   5) HDL-specified verification IP (VIP) (e.g., basic stimulus generators and checkers)

b) IP views—This is a list of different views (levels of description and/or languages) to describe the IP object. In IP-XACT v1.5, these views include:
   1) Design view: RTL Verilog or VHDL, flat or hierarchical components
   2) Simulation view: model views, targets, simulation directives, etc.
   3) Documentation view: standard, user guide, etc.

IP-XACT XML meta-data descriptions provide a standardized way of collecting much of the structural information contained in the file sets. IP-XACT also can contain the information that identifies the appropriate files included in a collection to be used for different parts of the design process.

[3]Available from the W3C Web site at http://www.w3.org/TR/2007/REC-soap12-part1-20070427/.

[4]Available from the W3C Web site at http://www.w3.org/TR/wsdl.

[5]SystemC is a registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries. Verilog is a registered trademark of Cadence Design Systems, Inc. in the United States and/or other jurisdictions. This information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE of these products. Equivalent products may be used if they can be shown to lead to the same results.

[6]See Footnote 5.

1.4 IP-XACT Enabled implementations

Complying with the rules outlined in this subclause allows the provider of tools, IP, or generators to class their products as *IP-XACT Enabled*. Conversely, any violation of these rules removes that naming right. This subclause first introduces the set of metrics for measuring the valid use of the specifications. It then specifies when those validity checks are performed by the various classes of products and providers: DEs, point tools, IPs, and generators.

a) Parse validity

   1) Parsing correctness: Ability to read all IP-XACT descriptions.

   2) Parsing completeness: Cannot require information that could be expressed in an IP-XACT format to be specified in a non-IP-XACT format. Processing of all information present in an IP-XACT document is not required.

b) Description validity

   1) Schema correctness: IP is described using XML files that conform to the IP-XACT schema.

   2) Usage completeness: Extensions to the IP-XACT schema shall only be used to express information that cannot otherwise be described in IP-XACT.

c) Semantic validity

   1) Semantic correctness: Adheres to the semantic interpretations of IP-XACT data described in this standard.

   2) Semantic completeness: Obeys all the semantic consistency rules (SCRs) described in Annex B.

These validity rules can be combined with the product class specific rules to cover the full IP-XACT enabled space. The following subclauses describe the rules a provider has to check to claim a product is IP-XACT Enabled.

An IP-XACT Enabled DE or point tool may read descriptions based on multiple versions of the IP-XACT schema. If the DE or point tool does provide this capability, the effect shall be as if all of the descriptions had been translated by the XSL Transform (XSLT) version 1.0,[7] which is provided with the IP-XACT release to convert descriptions from one version to the next. In addition, a DE or point tool may preserve information in the initial description for use outside of the scope of the IP-XACT specification.

1.4.1 Design environments

An IP-XACT Enabled DE:

a) Shall follow the parse validity requirements shown in 1.4.

b) Shall only create IP that is IP-XACT Enabled.

c) When modifying any existing IP-XACT descriptions, shall do so without losing any preexisting information. In particular, it shall preserve any vendor extension data included in the existing IP-XACT description.

d) Shall support the IP-XACT generator interfaces fully for interaction with underlying database.

e) Shall be able to invoke all IP-XACT Enabled generators.

XPATH version 1.0[8] support is required for DE-compliance.

[7]Available from the W3C Web site at http://www.w3.org/TR/1999/REC-xslt-19991116.
[8]Available from the W3C Web site at http://www.w3.org/TR/1999/REC-xpath-19991116.

1.4.2 Point tools

A point tool is a tool that has a particular rather than a general set of capabilities. In contrast to IP-XACT Enabled DE (see 1.4.1), an IP-XACT Enabled point tool:

a) Shall follow the parse validity requirements shown in 1.4.

b) Shall only create IP that is IP-XACT Enabled.

c) When modifying any existing IP-XACT descriptions, shall do so without losing any preexisting information. In particular, it shall preserve any vendor extension data included in the existing IP-XACT description.

1.4.3 IPs

An IP-XACT Enabled IP:

a) Shall have an IP-XACT description that follows the description and semantic validity requirements shown in 1.4.

b) Shall only use IP-XACT Enabled generators for any generators associated with this IP.

XML descriptions compliant to IP-XACT shall provide a namespace reference to the `index.xsd` schema file, not to any of the other files in the release.

1.4.4 Generators

An IP-XACT Enabled generator:

a) Shall only create IP that is IP-XACT Enabled.

b) When modifying any existing IP-XACT descriptions, shall do so without losing any preexisting information. In particular, it shall preserve any vendor extension data included in the existing IP-XACT description.

c) Shall be callable though the IP-XACT TGI (see Annex G).

d) Shall only communicate with the DE that invoked it through the IP-XACT TGI (see Annex G).

1.5 Conventions used

The conventions used throughout the document are included here.

IP-XACT is case-sensitive.

1.5.1 Visual cues (meta-syntax)

Bold shows required keywords and/or special characters, e.g., **addressSpace**. For the initial definitional use (per element), keywords are shown in **boldface-red text**, e.g, **bitsInLau** (see also: 1.6).

Bold italics shows group names or data types, e.g., ***nameGroup*** or ***boolean***. For definitions of types, see Annex D.

`Courier` shows examples, external command names, directories and files, etc., e.g., address `0x0` is on `D[31:0]`.

1.5.2 Notational conventions

The keywords *required, shall, shall not, should, should not, recommended, may,* and *optional* in this document are to be interpreted as described in the IETF Best Practices Document 14, RFC-2119 [B5].[9]

1.5.3 Syntax examples

Any syntax examples shown in this standard are for information only and are only intended to illustrate the use of such syntax.

1.5.4 Graphics used to document the schema

The W3C Web site[10] specifies the XML schema language used to define the IP-XACT XML schemas. Normative details for compliance to the IP-XACT standard are contained in the schema files. Within this document, pictorial representations of the information in the schema files *illustrate* the structure of the schema and *define* any constraints of the standard. With the exception of scope and visibility issues, the information in the figures and the schema files is intended to be identical. Where the figures and schema are in conflict, the XML schema file shall take precedence.[11]

1.5.4.1 Elements and attributes

The *element* is the fundamental building block on which this standard is based. An element may be either a *leaf element*, which is a container for information, or a *branch element*, which may contain further branch elements or leaf elements.

A leaf or branch element may also contain *attributes*. Attributes are containers for information within the containing element.

1.5.4.2 Types

A *type* is a designation of the format for the contents of an element or attribute. There are two different styles of types that can be defined. A type may be assigned to a leaf element or an attribute. This type is called a ***simpleType*** and defines the format of data that may be stored in this container. A type may also be assigned to a branch element. This type is called a ***complexType*** and defines further elements and attributes contained in the branch element.

1.5.4.3 Groups

A group is a collection of elements or attributes, which allow the same collection of items to be referenced consistently in many places. There are two different types of groups that can be defined. A *group* is a combination of leaf or branch elements; an ***attributeGroup***, a simple list of attributes. The names assigned to either group have no representation in the resulting description.

1.5.4.4 Namespace

Each element, attribute, type, or group has a name, which is preceded by a namespace and separated from the name by a colon (**:**). For the examples in 1.5.4.5, **xyz** is used as the namespace for all of the items

[9]The number in brackets correspond to those of the bibliography in Annex A.

[10]Available from the W3C Web site at http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/.

[11]The graphics for this document have been generated by taking "screen-shots" of the various files as they are displayed in Altova's XML environment XMLSpy®. XMLSpy is a registered trademark of Altova GmbH. This information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE of this product. Equivalent products may be used if they can be shown to lead to the same results.

whereas the standard uses **spirit**. Within the text of this standard, the namespace is not written when describing an item; it is only shown in examples.

1.5.4.5 Diagrams

The diagrams used throughout this standard graphically detail the organization the elements and attributes.

1.5.4.5.1 Elements and sequences

Figure 3 shows the sequence-compositor. At the left is a branch element, **element1**, with some descriptive text below. **element1** is connected to a sequence-compositor. The sequence-compositor defines the order the subelements appear in the branch element. **subElement1** shall appear first inside of **element1**. This is followed by **subElement2**, **subElement3**, **subElement4**, and **subElement5** before closing **element1**.

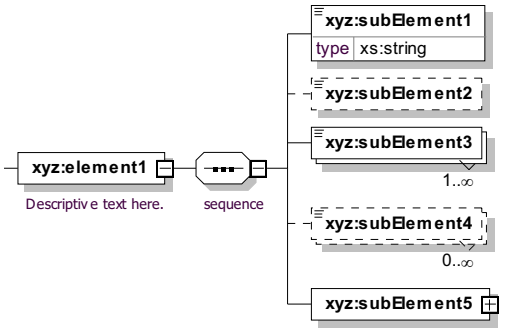

Figure 3—Sequence-compositor

a) **subElement1** is a mandatory element, as indicated by the solid line of the containing box. The type of the data contained in this element is set to ***string*** and it has a default value of `ip-xact` if the element is present, but left empty.

b) **subElement2** is an optional element, as indicated by the dashed-line of the containing box.

c) **subElement3** is an mandatory element that may appear multiple times, indicated by the double-solid line of the containing box. The number of times the element may appear is indicated by the range of the numbers listed below the element.

d) **subElement4** is an optional element that may appear multiple times, as indicated by the double-dashed line of the containing box. The number of times the element may appear is indicated by the range of the numbers listed below the element.

e) **subElement5** is an mandatory branch element that contains further elements inside, as indicated by the small plus sign (+) in the small box on the right.

Figure 4 shows variations of a sequence-compositor. **root1** is connected to an optional sequence-compositor, as indicated by the symbol being drawn with a dashed line. **element1** may appear first inside of **root1**; if it does, it shall be followed by **element2**. Each subelement is connected to a sequence-compositor.

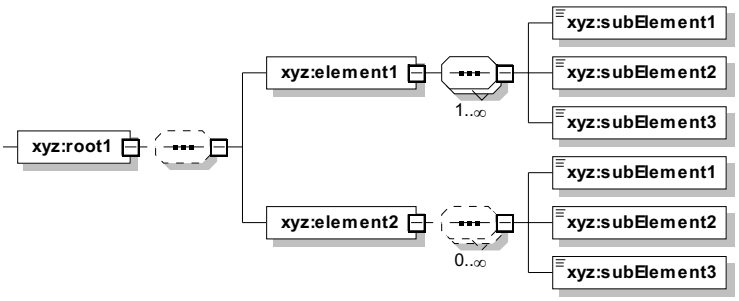

Figure 4—Sequence-compositor variations

— **element1** may contain one or more of the following sequences in the following order: **subElement1** and **subElement2** and **subElement3**. The number of times the sequence-compositor may appear is indicated by the range of the numbers listed below the symbol. If the range is greater than 1, the sequence-compositor symbol is drawn with double lines.

— **element2** is optional and may contain one or more of the following sequences in the following order: **subElement1** and **subElement2** and **subElement3**. The number of times the sequence-compositor may appear is indicated by the range of the numbers listed below the symbol. If the range starts at 0 and the maximum is greater then 1, the sequence-compositor symbol is drawn with double-dashed lines.

1.5.4.5.2 Elements and choices

Figure 5 shows the variations of the choice-compositor. **root** is connected to a choice-compositor. The choice-compositor specifies that one of the elements on the right side shall be chosen. **root** may contain one of the following: **element1**, **element2**, or **element3**. Each subelement is connected to a choice-compositor.

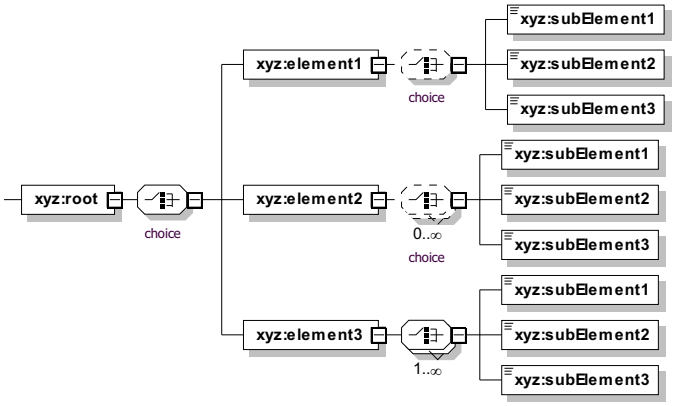

Figure 5—Choice-compositor variations

a) **element1** may contain one of the following: **subElement1**, **subElement2**, or **subElement3**, as indicated by the symbol being drawn with a dashed line.

b) **element2** may contain any (0 or more) of the following: **subElement1**, **subElement2**, or **subElement3** in any order. The number of times the choice-compositor may appear is indicated by

the range of the numbers listed below the symbol. If the range starts at 0, the choice-compositor is drawn with dashed lines.

c) **element3** may contain one or more of the following: **subElement1**, **subElement2**, or **subElement3** in any order. The number of times the choice-compositor may appear is indicated by the range of the numbers listed below the symbol. If the range is greater than 1, the choice-compositor is drawn with double lines.

1.5.4.5.3 Elements, attributes, groups, and attributeGroups

Figure 6 shows the use of attributes, groups, and attributeGroups. **element1** contains two attributes, shown in the tab shaped box labeled *attributes*. **attribute1** is optional, as indicated by the dashed containing box. **attribute1** also has a type defined of ***integer*** and a default value of 7 if the attribute is not present. **attribute2** is a required attribute, as indicated by the solid containing box, and is of type ***boolean*** with no default. The ordering in which **attribute1** and **attribute2** appear inside **element1** is irrelevant.

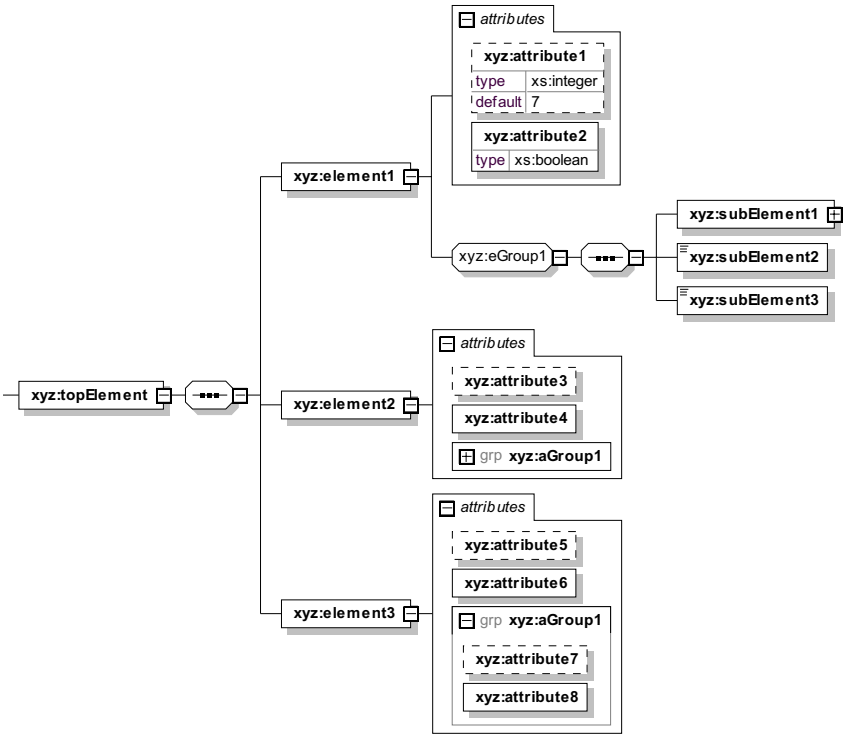

Figure 6—Attributes, groups, and attributeGroups

a) ***eGroup1*** is an element group inside **element1**. This group contains three subelements and the group symbol can be replaced by a solid line. The name of the group has no representation in the resulting output description. An element group can be optional, as indicated by a dashed outline (not shown) and it can also have a range, as indicated by numbers below the group symbol (not shown).

b) ***aGroup1*** is an ***attributeGroup*** inside **element2** and **element3**. This ***attributeGroup*** contains two attributes, **attribute7** and **attribute8**. Inside **element2**, the ***attributeGroup*** is shown in its collapsed form, as indicated by the small plus sign (+) inside the small box. Inside **element3** the ***attributeGroup*** is shown in it expanded form, as indicated by the small minus sign (−) inside the small box. **element2** contains four attributes: **attribute3**, **attribute4**, **attribute7**, and **attribute8**. **element3** also contains four attributes: **attribute5**, **attribute6**, **attribute7**, and **attribute8**. The name of the ***attributeGroup*** has no representation in the resulting description.

1.5.4.5.4 Wildcards

Figure 7 shows the use of wildcards. A *wildcard* is depicted by the rounded box with the **any ##any** text. Wildcards indicate that any well-formed attribute or element may be inserted into the containing element.

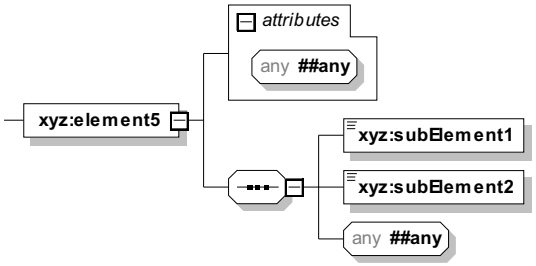

Figure 7—Wildcards

1.6 Use of color in this standard

This standard uses a minimal amount of color to enhance readability. The coloring is not essential and does not affect the accuracy of this standard when viewed in pure black and white. The places where color is used are the following:

— Cross references that are hyperlinked to other portions of this standard are shown in underlined-blue text (hyperlinking works when this standard is viewed interactively as a PDF file).

— Syntactic keywords and tokens in the formal language definitions are shown in **boldface-red text**.

1.7 Contents of this standard

The organization of the remainder of this standard is as follows:

— Clause 2 provides references to other applicable standards that are assumed or required for this standard.

— Clause 3 defines terms, acronyms, and abbreviations used throughout the different specifications contained in this standard.

— Clause 4 defines the interoperability use model.

— Clause 5 defines the bus and abstraction definitions.

— Clause 6 defines the component and interconnect models.

— Clause 7 defines the designs and their connections.

— Clause 8 defines the abstractor model between abstraction definitions.

— Clause 9 defines the generator chain.

— Clause 10 defines the design and generator chain configuration.

— Clause 11 defines addressing and data visibility.

— Annexes. Following Clause 11 are a series of annexes.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

Simple Object Access Protocol (SOAP) version 1.2 specification, available from the W3C Web site at http://www.w3.org/TR/2007/REC-soap12-part1-20070427/.

Web Services Description Language (WSDL) version 1.1 specification, available from the W3C Web site at http://www.w3.org/TR/wsdl.

XML schema specification, available from the W3C Web site at http://www.w3.org/TR/2004/REC-xmlschema-0-20041028; http://www.w3.org/TR/2004/REC-xmlschema-1-20041028; http://www.w3.org/TR/2004/PER-xmlschema-2-20040318.

XML version 1.0 specification, available from the W3C Web site at http://www.w3.org/TR/2000/REC-xml-20001006.

XPATH version 1.0 specification, available from the W3C Web site at http://www.w3.org/TR/1999/REC-xpath-19991116.

XSLT version 1.0 specification, available from the W3C Web site at http://www.w3.org/TR/1999/REC-xslt-19991116.