Accellera Vendor Extensions for IEEE 1685-2009

Version 1.0

September 2013

Accellera Systems Initiative, 1370 Trancas Street #163, Napa, CA 94558, USA.

Notices

Accellera Systems Initiative Standards documents are developed within Accellera Systems Initiative and the Technical Committees and Working Groups of Accellera Systems Initiative, Inc. Accellera Systems Initiative develops its standards through a consensus development process, approved by its members and board of directors, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of Accellera and serve without compensation. While Accellera Systems Initiative administers the process and establishes rules to promote fairness in the consensus development process, Accellera Systems Initiative does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an Accellera Systems Initiative Standard is wholly voluntary. Accellera Systems Initiative disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other Accellera Systems Initiative Standard document.

Accellera Systems Initiative does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or suitability for a specific purpose, or that the use of the material contained herein is free from patent infringement. Accellera Systems Initiative Standards documents are supplied "**AS IS**."

The existence of an Accellera Systems Initiative Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of an Accellera Systems Initiative Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change due to developments in the state of the art and comments received from users of the standard. Every Accellera Systems Initiative Standard is subjected to review periodically for revision and update. Users are cautioned to check to determine that they have the latest edition of any Accellera Systems Initiative Standard.

In publishing and making this document available, Accellera Systems Initiative is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is Accellera Systems Initiative undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other Accellera Systems Initiative Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of Accellera Systems Initiative, Accellera Systems Initiative will initiate reasonable action to prepare appropriate responses. Since Accellera Systems Initiative Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, Accellera Systems Initiative and the members of its Technical Committees and Working Groups are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments for revision of Accellera Systems Initiative Standards are welcome from any interested party, regardless of membership affiliation with Accellera Systems Initiative. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Accellera Systems Initiative
1370 Trancas Street #163
Napa, CA 94558
USA

Note: 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. Accellera Systems Initiative shall not be responsible for identifying patents for which a license may be required by an Accellera Systems Initiative standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Accellera Systems Initiative is the sole entity that may authorize the use of Accellera Systems Initiative -owned certification marks and/or trademarks to indicate compliance with the materials set forth herein.

Authorization to photocopy portions of any individual standard for internal or personal use must be granted by Accellera Systems Initiative Inc., provided that permission is obtained from and any required fee, if any, is paid to Accellera Systems Initiative. To arrange for authorization please contact Lynn Bannister, Accellera Systems Initiative, 1370 Trancas Street #163, Napa, CA 94558, phone (707) 251-9977, e-mail lynn@accellera.org. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained from Accellera Systems Initiative.

Suggestions for improvements to the IP-XACT Recommended Extensions are welcome. They should be sent to the group's email Reflector:

ip-xact@lists.accellera.org

The current IP-XACT Technical Committee web page is:

www.accellera.org/activities/committees/ip-xact

Participants

The following members of the IP-XACT Extensions Working Group actively contributed to the recommended extensions.

- Erwin de Kock (Chair)
- Emmanuel Vaumorin (Vice Chair)
- Grégoire Avot
- Martin Barnasconi
- Olivier Florent
- Chris Neely

The following companies of the IP-XACT Extensions Working Group voted on the recommended extensions.

- Magillem Design Services
- NXP Semiconductors
- STMicroelectronics
- Xilinx, Inc.

The IP-XACT Extensions Working Group members gratefully acknowledge the contributions of the following IP-XACT Compliance Working Group members concerning schema authoring and schema validation.

- Edwin Dankert
- Stéphane Guntz

Table of Contents

1. Overview

This section explains the scope and purpose of this standard. Furthermore, it gives an overview of the basic concepts and summarizes its contents.

1.1. Scope

This standard describes XML schemas for IP-XACT vendor extensions that have been defined by the Accellera IP-XACT Technical Committee. These XML schemas are called Accellera Vendor Extensions.

1.2. Purpose

This standard enables the use of the IP-XACT standard IEEE Std 1685™-2009 in domains that are not supported by the IP-XACT standard yet. More specifically, this standard enables the use of the IP-XACT standard in analog and mixed signal flows, physical design planning flows, and power flows.

1.3. Concepts

Accellera vendor extensions make use of accellera extension containers. An Accellera extension container is an XML element that can occur in an IP-XACT vendor extension. The containers are defined in a generic unversioned namespace to enable generic processing of standard extensions by IP-XACT enabled implementations; see Section 1.4. The sub-elements of the containers are defined in domain-specified versioned namespaces such that domain-specific tools do not have to process information from other domains.

1.4. IP-XACT Enabled Implementations

As described in IEEE Std 1685™-2009, Section 1.4.1, when modifying any existing IP-XACT descriptions, IP-XACT enabled design environments shall do so without losing any preexisting information. In particular, they shall preserve any vendor extension data included in the existing IP-XACT description.

In addition, IP-XACT enabled implementations that support Accellera vendor extensions shall not corrupt IP-XACT descriptions containing Accellera vendor extensions. More specifically, such implementations shall process name references implemented by the accellera:nameRef and accellera:viewNameRef elements correctly and avoid dangling name references.

1.5. Outline

Section 2 defines the XML schema for the “accellera” namespace to describe the organization of Accellera extensions containers. Section 3 defines the XML schema for the “accellera-core” namespace to describe meta-data that is relevant in multiple domains. Section 4 defines the XML schema for the “accellera-ams” namespace to describe meta-data that is relevant in the analog and mixed signal domain. Section 5 defines the XML schema for the “accellera-pdp” namespace to describe meta-data that is relevant in physical design planning domain. Section 6 defines the XML schema for the “accellera-power” namespace to describe meta-data that is relevant in the powerl domain. Section7 describes the semantic consistency rules that need to be obeyed by vendor extensions in order to be valid Accellera vendor extensions.

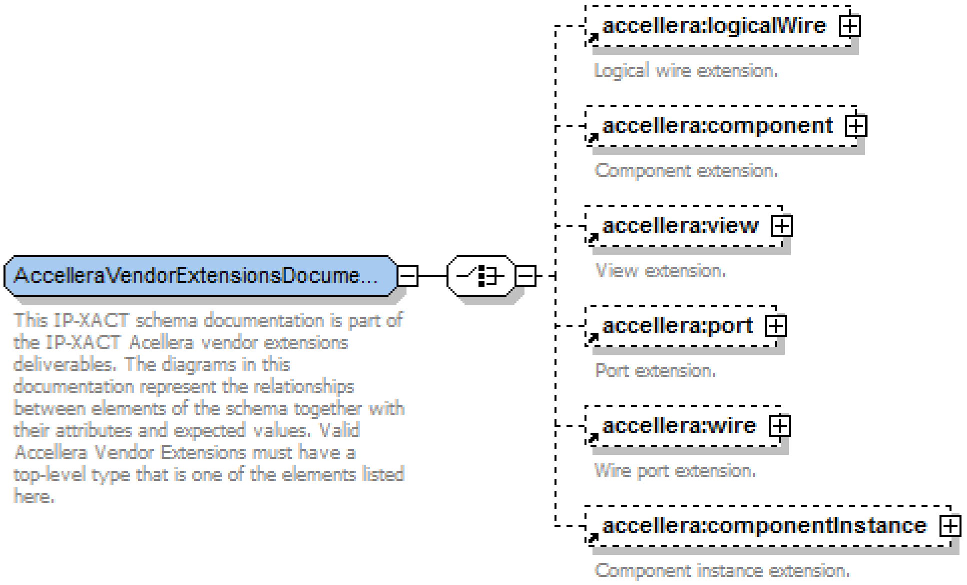
2. Accellera vendor extensions containers

2.1. Containers

An *Accellera vendor extensions container* is a container for Accellera vendor extensions meta-data. All containers are defined in the namespace “accellera”. This namespace is not versioned to enable generic extensions support from IP-XACT enabled design environments and tools. The container name is identical to the spirit element name that it extends, except for the element names that exist in abstraction definitions which conflict with element names in components such as ports, port, wire, and transactional. The containers for these elements in abstraction definitions are prefixed with “logical” in order to distinguish them from the containers in components. In IEEE Std 1685™-2009, some spirit elements have a vendor extension and some spirit elements do not have a vendor extension sub-element. For spirit elements that have a vendor extension sub-element, the container element must be located in that vendor extension sub-element. For spirit elements that do not have a vendor extension sub-element, the container element must be located in the nearest encapsulating spirit element that has a vendor extension sub-element.

2.1.1. Schema

The following schema defines the Accellera vendor extensions container elements.



This IP-XACT schema documentation is part of the IP-XACT Acellera vendor extensions deliverables. The diagrams in this documentation represent the relationships between elements of the schema together with their attributes and expected values. Valid Accellera Vendor Extensions must have a top-level type that is one of the elements listed here.

2.1.2. Description

Each element of the **AcceleraVendorExtensionsExtensionsDocumentTypes** defines a container:

a) **accellera:logicalWire** (optional) specifies meta-data for wire ports in an abstraction definition.
b) **accellera:component** (optional) specifies meta-data for a component.
c) **accellera:view** (optional) specifies meta-data for a view in a component.
d) **accellera:port** (optional) specifies meta-data for a port in a component.
e) **accellera:wire** (optional) specifies meta-data for a wire port in a component.
f) **accellera:componentInstance** (optional) specifies meta-data for a component instance in a design.

2.1.3. Example

This is an example of a component model wire port containing a standard extensions container. The declarations of the domain-specific namespaces are required only if they are actually used in the document.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<spirit:componentxsi:schemaLocation="
http://www.spiritconsortium.org/XMLSchema/SPIRIT/1685-2009
```

```
http://www.accellera.org/XMLSchema/SPIRIT/1685-2009/index.xsd
http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE
http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE-1.0/index.xsd"
xmlns:spirit="http://www.spiritconsortium.org/XMLSchema/SPIRIT/1685-2009"
xmlns:accellera="http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE"
xmlns:accellera-core="http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE/CORE-1.0"
xmlns:accellera-ams="http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE/AMS-1.0"
xmlns:accellera-pdp="http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE/PDP-1.0"
xmlns:accellera-power="http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE/POWER-
1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <spirit:vendor>accellera.org</spirit:vendor>
  <spirit:library>ipxact</spirit:library>
  <spirit:name>component</spirit:name>
  <spirit:version>1.0</spirit:version>
  <spirit:model>
    <spirit:ports>
      <spirit:port>
        <spirit:name>myport</spirit:name>
        <spirit:wire>
          <spirit:direction>in</spirit:direction>
        </spirit:wire>
        <spirit:vendorExtensions>
          <accellera:wire>
            …
          </accellera:wire>
        </spirit:vendorExtensions>
      </spirit:port>
    </spirit:ports>
  </spirit:model>
</spirit:component>
```

The next example shows a design component instance containing a standard extensions container.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<spirit:design xsi:schemaLocation="
http://www.spiritconsortium.org/XMLSchema/SPIRIT/1685-2009
http://www.accellera.org/XMLSchema/SPIRIT/1685-2009/index.xsd
http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE
http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE-1.0/index.xsd"
xmlns:spirit="http://www.spiritconsortium.org/XMLSchema/SPIRIT/1685-2009"
xmlns:accellera="http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE"
xmlns:accellera-core="http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE/CORE-1.0"
xmlns:accellera-ams="http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE/AMS-1.0"
xmlns:accellera-pdp="http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE/PDP-1.0"
xmlns:accellera-power="http://www.accellera.org/XMLSchema/SPIRIT/1685-2009-VE/POWER-
1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <spirit:vendor>accellera.org</spirit:vendor>
  <spirit:library>ipxact</spirit:library>
  <spirit:name>design</spirit:name>
  <spirit:version>1.0</spirit:version>
  <spirit:componentInstances>
    <spirit:componentInstance>
      <spirit:instanceName>i_component</spirit:instanceName>
      <spirit:componentRef spirit:version="1.0" spirit:name="component"
        spirit:library="ipxact" spirit:vendor="accellera.org"/>
      <spirit:vendorExtensions>
        <accellera:componentInstance>
          …
        </accellera:componentInstance>
      </spirit:vendorExtensions>
    </spirit:componentInstance>
  </spirit:componentInstances>
</spirit:design>
```

2.2. Name references

In containers it is possible to refer to **spirit:name** elements by using the **accellera:nameRef** element. Please see Section 5.2.3 for an example.

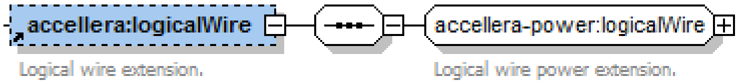
2.3. View name references

In containers it is possible to refer to **spirit:name** elements inside view elements by using the **accellera:viewNameRef** element. Please see Section 3.2.3.3 for an example.

2.4. Logical wire container

2.4.1. Schema

The following schema defines the contents of the logical wire container.



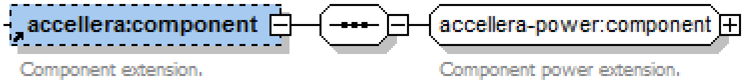
2.4.2. Description

The logical wire container contains Accellera vendor extensions in the power namespace.

2.5. Component container

2.5.1. Schema

The following schema defines the contents of the component container.



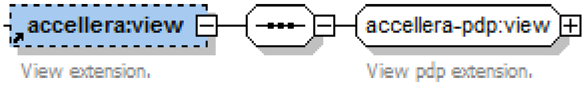
2.5.2. Description

The component container contains Accellera vendor extensions in the power namespace.

2.6. View container

2.6.1. Schema

The following schema defines the contents of the view container.



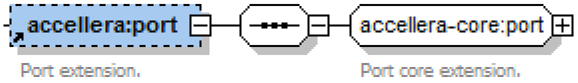
2.6.2. Description

The view container contains Accellera vendor extensions in the physical design planning namespace.

2.7. Port container

2.7.1. Schema

The following schema defines the contents of port container.



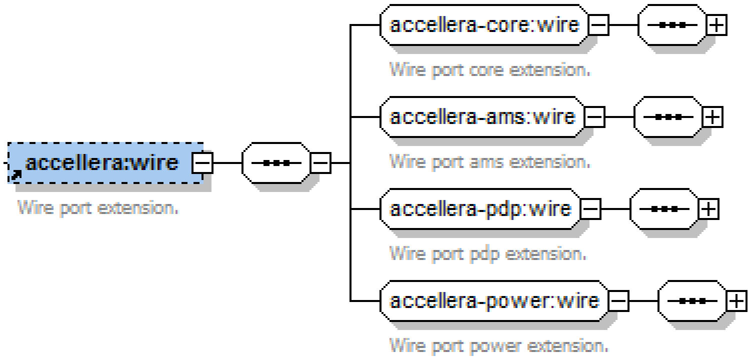
2.7.2. Description

The port container contains Accellera vendor extensions in the core namespace.

2.8. Wire container

2.8.1. Schema

The following schema defines the contents of the wire container.



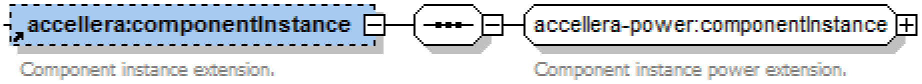
2.8.2. Description

The wire container contains Accellera vendor extensions in the core, analog and mixed signal, physical design planning, and power namespaces.

2.9. Component instance container

2.9.1. Schema

The following schema defines the contents of the component instance container.


accellera:componentInstance
Component instance extension.
accellera-power:componentInstance
Component instance power extension.

2.9.2. Description

The component instance container contains Accellera vendor extensions in the power namespace.

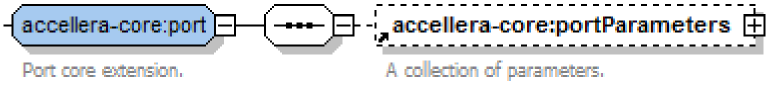
3. Core extensions

The core extensions are defined in the “accellera-core” namespace. The purpose of core extensions is to describe vendor extensions meta-data that is shared between domain-specific namespaces.

3.1. Port extension

3.1.1. Schema

The following schema defines the contents of the core port container.



3.1.2. Description

The core port extension contains the following elements.

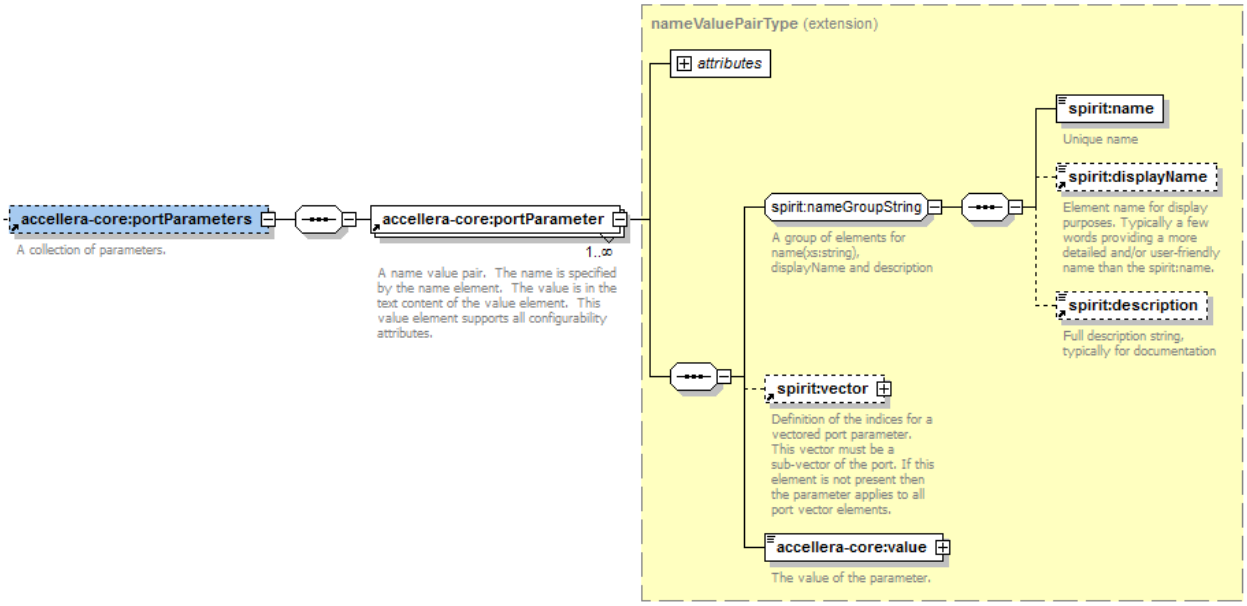
a) **accellera-core:portParameters** (optional) is a container for one or more port parameters; see Section 3.1.3.

3.1.3. Port parameter

A port parameter differs from a spirit parameter element because it contains a vector element and it supports parameter value attributes unit and prefix.

3.1.3.1. Schema

The following schema defines the contents of the port parameter element.


nameValuePairType (extension)
attributes
accellera-core:portParameters
A collection of parameters.
accellera-core:portParameter
1..∞
A name value pair. The name is specified by the name element. The value is in the text content of the value element. This value element supports all configurability attributes.
spirit:nameGroupString
A group of elements for name(xs:string), displayName and description
spirit:name
Unique name
spirit:displayName
Element name for display purposes. Typically a few words providing a more detailed and/or user-friendly name than the spirit:name.
spirit:description
Full description string, typically for documentation
spirit:vector
Definition of the indices for a vectored port parameter. This vector must be a sub-vector of the port. If this element is not present then the parameter applies to all port vector elements.
accellera-core:value
The value of the parameter.

3.1.3.2. Description

a) **spirit:nameGroup** group containing **name**, **displayName**, and **description**. The name together with the vector range shall be unique within the containing portParameters element.

b) **spirit:vector** (optional) determines on which port elements the parameter shall be applied.

c) **accellera-core:value** (mandatory) specifies the parameter value; see Section 3.3.

3.1.3.3. Example

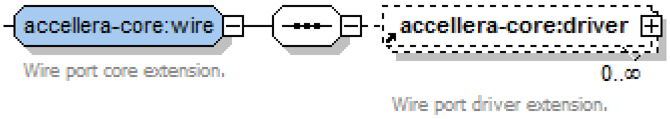
This is an example of a port extension containing a port parameter.

```
<accellera:port>
  <accellera-core:portParameters>
    <accellera-core:portParameter>
      <spirit:name>Voltage</spirit:name>
      <accellera-core:value>1.3</accellera-core:value>
    </accellera-core:portParameter>
  </accellera-core:portParameters>
</accellera:port>
```

3.2. Wire extension

3.2.1. Schema

The following schema defines the contents of the core wire container.



3.2.2. Description

The core wire extension contains the following elements.

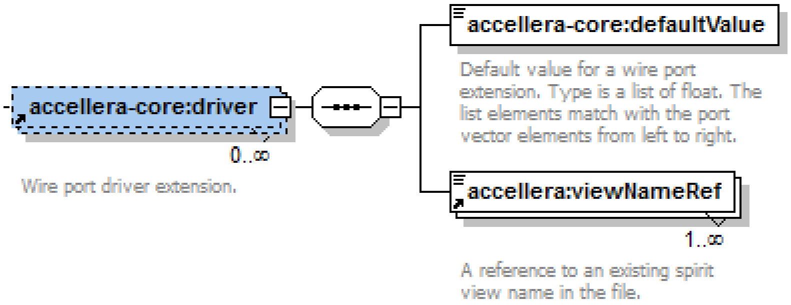
a) **accellera-core:driver** (optional) defines a driver for the wire port; see Section 3.2.3.

3.2.3. Driver

A driver differs from a spirit driver because it supports other values than non-negative integers.

3.2.3.1. Schema

The following schema defines the contents of the driver element.


accellera-core:driver
0..∞
Wire port driver extension.
accellera-core:defaultValue
Default value for a wire port extension. Type is a list of float. The list elements match with the port vector elements from left to right.
accellera:viewNameRef
1..∞
A reference to an existing spirit view name in the file.

3.2.3.2. Description

The driver element contains the following elements.

a) **accellera-core:defaultValue** (mandatory) representing a list of floats. Elements of a list match with the port elements from left to right in a vector.

a) **accellera:viewNameRef** (mandatory)) indicates the view or views in which this driver applies. Multiple views can use the same set of type properties by specifying multiple **viewNameRef** elements. The **viewNameRef** shall match a **view name** in the containing object. The **viewNameRef** element is of type ***NMTOKEN***.

3.2.3.3. Example

This is an example of a driver extension.

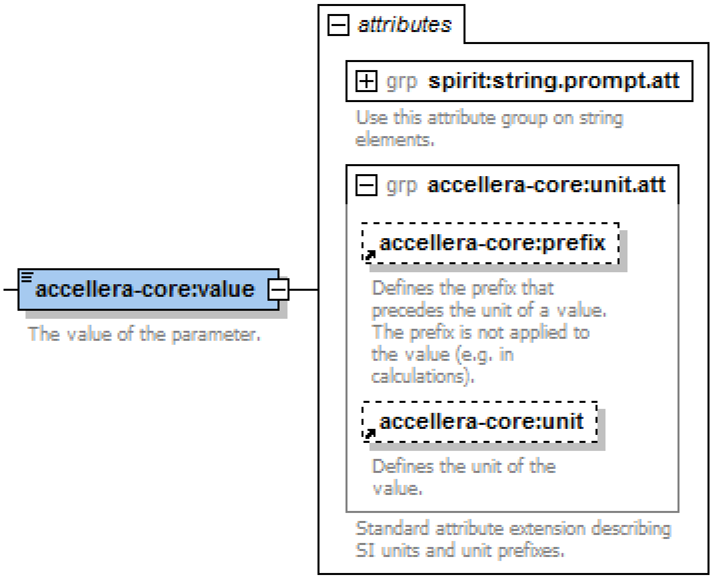
```
<spirit:port>
  <spirit:name>gnds</spirit:name>
  <spirit:wire>
    <spirit:direction>in</spirit:direction>
    <spirit:driver>
      <spirit:defaultValue>1</spirit:defaultValue>
    </spirit:driver>
  </spirit:wire>
  <spirit:vendorExtensions>
    <accellera:wire>
      <accellera-core:driver>
        <accellera-core:defaultValue>0.7</accellera-core:defaultValue>
        <accellera:viewNameRef>functional-ams</accellera:viewNameRef>
      </accellera-core:driver>
    </accellera:wire>
  </spirit:vendorExtensions>
</spirit:port>
```

3.3. Value

A value differs from a spirit value because it supports two additional attributes prefix and unit.

3.3.1. Schema

The following schema defines the contents of the value element.


attributes
grp spirit:string.prompt.att
Use this attribute group on string elements.
grp accellera-core:unit.att
accellera-core:prefix
Defines the prefix that precedes the unit of a value. The prefix is not applied to the value (e.g. in calculations).
accellera-core:unit
Defines the unit of the value.
Standard attribute extension describing SI units and unit prefixes.
accellera-core:value
The value of the parameter.

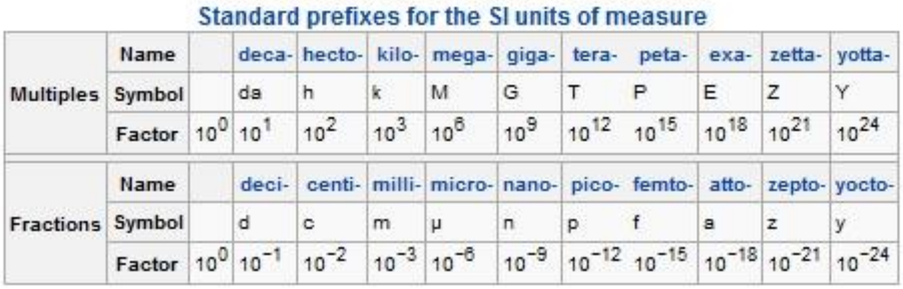
3.3.2. Description

The element **accellera-core:value** is configurable with attributes from spirit:string.prompt.att. In addition it has two optional attributes

a) **accellera-core:prefix** which can take the values in Table 1.
b) **accellera-core:unit** which can take the relevant units in Table 2 and Table 3, respectively, i.e., second, ampere, Kelvin, hertz, joule, watt, coulomb, volt, farad, ohm, Siemens, henry, Celsius.

Note that both **prefix** and **unit** can be used for documentation purposes only. The prefix attribute is not used to normalize the value to the base unit!

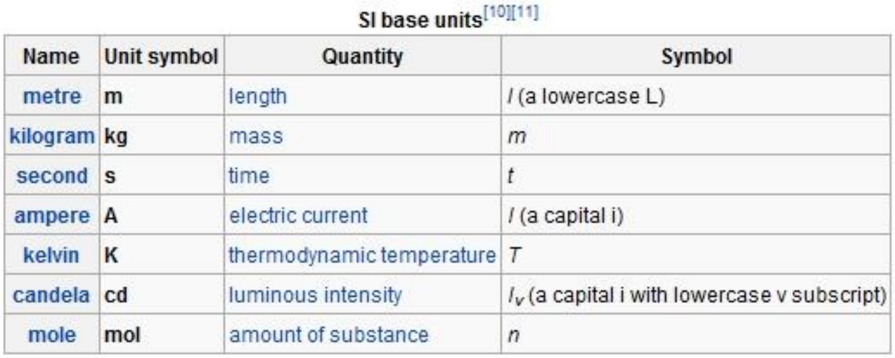
Table 1: Standard prefixes for the SI units [source: Wikipedia].



Standard prefixes for the SI units of measure

Multiples	**Name**		deca-	hecto-	kilo-	mega-	giga-	tera-	peta-	exa-	zetta-	yotta-
	Symbol		da	h	k	M	G	T	P	E	Z	Y
	Factor	10^0	10^1	10^2	10^3	10^6	10^9	10^{12}	10^{15}	10^{18}	10^{21}	10^{24}
Fractions	**Name**		deci-	centi-	milli-	micro-	nano-	pico-	femto-	atto-	zepto-	yocto-
	Symbol		d	c	m	µ	n	p	f	a	z	y
	Factor	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-6}	10^{-9}	10^{-12}	10^{-15}	10^{-18}	10^{-21}	10^{-24}

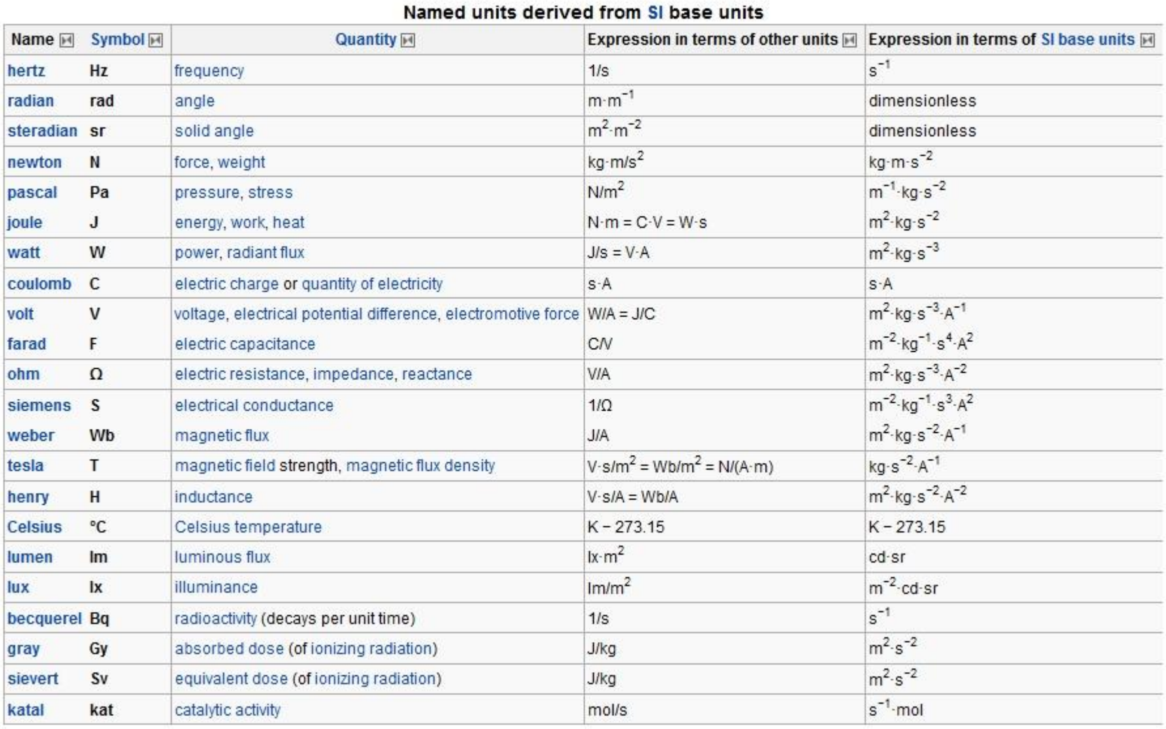
Table 2: SI base units [source: Wikipedia].



SI base units[10][11]

Name	Unit symbol	Quantity	Symbol
metre	**m**	length	*l* (a lowercase L)
kilogram	**kg**	mass	*m*
second	**s**	time	*t*
ampere	**A**	electric current	*I* (a capital i)
kelvin	**K**	thermodynamic temperature	*T*
candela	**cd**	luminous intensity	I_v (a capital i with lowercase v subscript)
mole	**mol**	amount of substance	*n*

Table 3: Units derived from SI base units [source: Wikipedia].



Named units derived from SI base units

Name	Symbol	Quantity	Expression in terms of other units	Expression in terms of SI base units
hertz	Hz	frequency	1/s	s^{-1}
radian	rad	angle	$m \cdot m^{-1}$	dimensionless
steradian	sr	solid angle	$m^2 \cdot m^{-2}$	dimensionless
newton	N	force, weight	$kg \cdot m/s^2$	$kg \cdot m \cdot s^{-2}$
pascal	Pa	pressure, stress	N/m^2	$m^{-1} \cdot kg \cdot s^{-2}$
joule	J	energy, work, heat	N·m = C·V = W·s	$m^2 \cdot kg \cdot s^{-2}$
watt	W	power, radiant flux	J/s = V·A	$m^2 \cdot kg \cdot s^{-3}$
coulomb	C	electric charge or quantity of electricity	s·A	s·A
volt	V	voltage, electrical potential difference, electromotive force	W/A = J/C	$m^2 \cdot kg \cdot s^{-3} \cdot A^{-1}$
farad	F	electric capacitance	C/V	$m^{-2} \cdot kg^{-1} \cdot s^4 \cdot A^2$
ohm	Ω	electric resistance, impedance, reactance	V/A	$m^2 \cdot kg \cdot s^{-3} \cdot A^{-2}$
siemens	S	electrical conductance	1/Ω	$m^{-2} \cdot kg^{-1} \cdot s^3 \cdot A^2$
weber	Wb	magnetic flux	J/A	$m^2 \cdot kg \cdot s^{-2} \cdot A^{-1}$
tesla	T	magnetic field strength, magnetic flux density	$V \cdot s/m^2 = Wb/m^2 = N/(A \cdot m)$	$kg \cdot s^{-2} \cdot A^{-1}$
henry	H	inductance	V·s/A = Wb/A	$m^2 \cdot kg \cdot s^{-2} \cdot A^{-2}$
Celsius	°C	Celsius temperature	K − 273.15	K − 273.15
lumen	lm	luminous flux	$lx \cdot m^2$	cd·sr
lux	lx	illuminance	lm/m^2	$m^{-2} \cdot cd \cdot sr$
becquerel	Bq	radioactivity (decays per unit time)	1/s	s^{-1}
gray	Gy	absorbed dose (of ionizing radiation)	J/kg	$m^2 \cdot s^{-2}$
sievert	Sv	equivalent dose (of ionizing radiation)	J/kg	$m^2 \cdot s^{-2}$
katal	kat	catalytic activity	mol/s	$s^{-1} \cdot mol$

3.3.3. Example

This is an example of a port extension containing a port parameter with a value using the unit and prefix attributes.

```
<accellera:port>
  <accellera-core:portParameters>
    <accellera-core:portParameter>
      <spirit:name>Voltage</spirit:name>
      <accellera-core:value accellera-core:unit="volt" accellera-core:prefix="kilo">
        1.3
      </accellera-core:value>
    </accellera-core:portParameter>
  </accellera-core:portParameters>
</accellera:port>
```

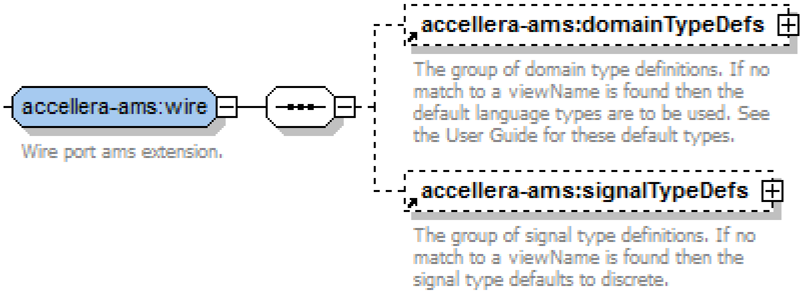
4. Analog and mixed signal extensions

The analog and mixed signal extensions are defined in the “accellera-ams” namespace. The purpose of analog and mixed signal extensions is to describe vendor extensions meta-data that enables integration of analog IPs as well as the mixture of analog and digital IPs.

4.1. Wire extension

4.1.1. Schema

The following schema defines the contents of the analog and mixed signal wire container.


accellera-ams:wire
Wire port ams extension.
accellera-ams:domainTypeDefs
The group of domain type definitions. If no match to a viewName is found then the default language types are to be used. See the User Guide for these default types.
accellera-ams:signalTypeDefs
The group of signal type definitions. If no match to a viewName is found then the signal type defaults to discrete.

4.1.2. Description

The analog and mixed signal wire extension contains the following elements.

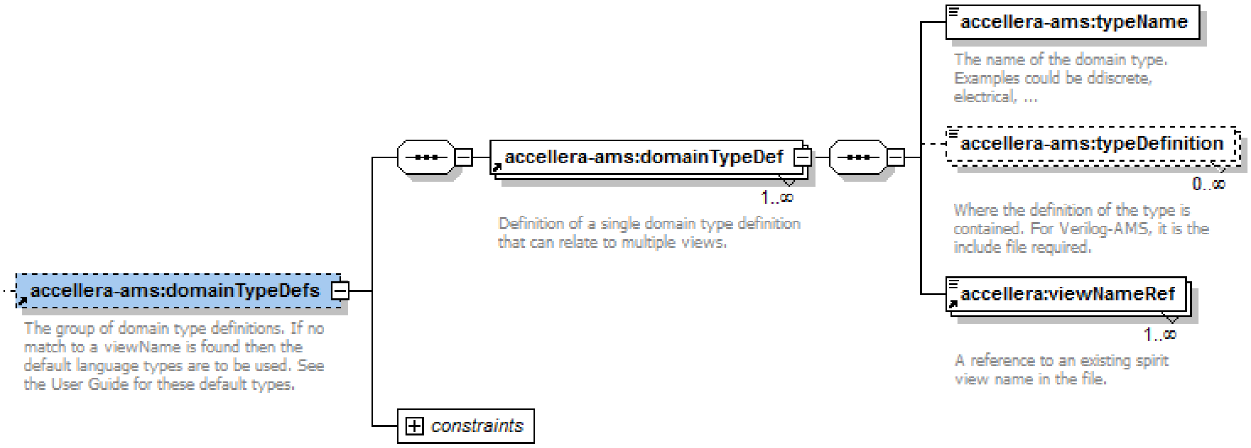
a) **accellera-ams:domainTypeDefs** (optional) describes the domain type properties for a port per view of a component or abstractor. There can be an unbounded series of **accellera-ams:domainTypeDef**s defined for each port, allowing the type properties to be defined differently for each view; **domainTypeDef** is defined in Section 4.1.3.

b) **accellera-ams:signalTypeDefs** (optional) describes the signal type properties for a port per view of a component or abstractor. There can be an unbounded series of **accellera-ams:signalTypeDef**s defined for each port, allowing the type properties to be defined differently for each view; **signalTypeDef** is defined in Section 4.1.4.

4.1.3. Domain type definition

4.1.3.1. Schema

The following schema defines the contents of a domain type definition.


accellera-ams:typeName
The name of the domain type. Examples could be ddiscrete, electrical, ...
accellera-ams:domainTypeDef
1..∞
Definition of a single domain type definition that can relate to multiple views.
accellera-ams:typeDefinition
0..∞
Where the definition of the type is contained. For Verilog-AMS, it is the include file required.
accellera-ams:domainTypeDefs
The group of domain type definitions. If no match to a viewName is found then the default language types are to be used. See the User Guide for these default types.
accellera:viewNameRef
1..∞
A reference to an existing spirit view name in the file.
constraints

4.1.3.2. Description

A domain type definition contains the following elements.

b) **accellera-ams:typeName** (mandatory) defines the name of the domain type for the port. For VHDL, some typical values would be ddiscrete and electrical. The domainTypeName element is of type ***string***.

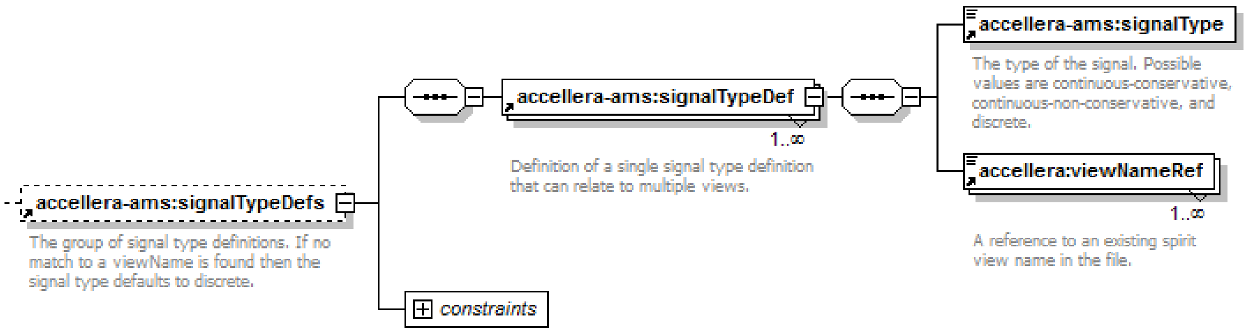
c) **accellera-ams:typeDefinition** (optional) contains a language-specific reference to where the given type is actually defined. There can be multiple **typeDefinition**s for each port. The **typeDefinition** element is of type ***string***.

d) **accellera:viewNameRef** (mandatory) indicates the view or views in which this type definition applies. Multiple views can use the same set of type properties by specifying multiple **viewNameRef** elements. The **viewNameRef** shall match a **view name** in the containing object. The **viewNameRef** element is of type ***NMTOKEN***.

4.1.4. Signal type definition

4.1.4.1. Schema

The following schema defines the contents of a domain type definition.


accellera-ams:signalTypeDefs
The group of signal type definitions. If no match to a viewName is found then the signal type defaults to discrete.
accellera-ams:signalTypeDef
1..∞
Definition of a single signal type definition that can relate to multiple views.
accellera-ams:signalType
The type of the signal. Possible values are continuous-conservative, continuous-non-conservative, and discrete.
accellera:viewNameRef
1..∞
A reference to an existing spirit view name in the file.
constraints

4.1.4.2. Description

A signal type definition contains the following elements.

a) **accellera-ams:signalType** (mandatory) defines the signal type for the port. **Continuous-conservative** is a signal which is continuous in time and quantity, where the quantity represents voltage and current to capture the conservative behavior of an electrical network (i. e. Kirchhoff's Laws apply), and which has no direction. **Continuous-non-conservative** is a directed signal which is continuous in time and quantity, where the quantity represents either a voltage or a current (often called signal flow). In this case it only captures the non-conservative behavior of a set of interconnected components. **Discrete** is a directed sampled signal which is time-discrete and value-continuous (the value may also be quantified), where the value represents either a voltage or a current (other representations are possible also). It captures the non-conservative behavior of a set of interconnected components.

b) **accellera:viewNameRef** (mandatory) indicates the view or views in which this type definition applies. Multiple views can use the same set of type properties by specifying multiple **viewNameRef** elements. The **viewNameRef** shall match a **view name** in the containing object. The **viewNameRef** element is of type ***NMTOKEN***.

Table 4 shows the mapping of the three signal types onto VHDL, Verilog, and SystemC AMS types.

Table 4: Mapping of signal types onto data types in various HDLs.

signalType	VHDL-AMS	Verilog-AMS	SystemC-AMS
continuous conservative	terminal outp : electrical;	electrical outp;	sca_eln::sca_terminal outp;1)
continuous non-conservative	quantity outp : out real;	sf_voltage outp; 2)	sca_lsf::sca_out outp; 1)
discrete	signal outp : out real;	wreal outp; ddiscrete outp; 3)	sca_tdf::sca_out<double> outp;

Notes:

1. SystemC-AMS 1.0 only supports linear electrical networks (ELN) and linear signal flow (LSF).
2. voltage is defined as: `discipline sf_voltage; potential Voltage; enddiscipline`. The discipline sf_voltage declaration must be described in a **domainTypeDef**.
3. ddiscrete is defined as: `discipline ddiscrete; domain discrete; enddiscipline` The discipline ddiscrete declaration must be described in a **domainTypeDef**.

4.1.5. Example

This is an example of domainTypeDefs and signalTypeDefs.

```
<spirit:port>
  <spirit:name>myport</spirit:name>
  <spirit:wire>
    <spirit:direction>in</spirit:direction>
  </spirit:wire>
  <spirit:vendorExtensions>
    <accellera:wire>
      <accellera-ams:domainTypeDefs>
        <accellera-ams:domainTypeDef>
          <accellera-ams:domainTypeName>electrical</accellera-ams:domainTypeName>
          <accellera-ams:typeDefinition>disciplines.vams
          </accellera-ams:typeDefinition>
          <accellera:viewNameRef>functional-ana</accellera:viewNameRef>
        </accellera-ams:domainTypeDef>
      </accellera-ams:domainTypeDefs>
      <accellera-ams:signalTypeDefs>
        <accellera-ams:signalTypeDef>
          <accellera-ams:signalType>continuous-conservative</accellera-ams:signalType>
          <accellera:viewNameRef>functional-ana</accellera:viewNameRef>
        </accellera-ams:signalTypeDef>
        <accellera-ams:signalTypeDef>
          <accellera-ams:signalType>discrete</accellera-ams:signalType>
          <accellera:viewNameRef>functional-sca-tdf</accellera:viewNameRef>
        </accellera-ams:signalTypeDef>
      </accellera-ams:signalTypeDefs>
    </accellera:wire>
  </spirit:vendorExtensions>
</spirit:port>
```

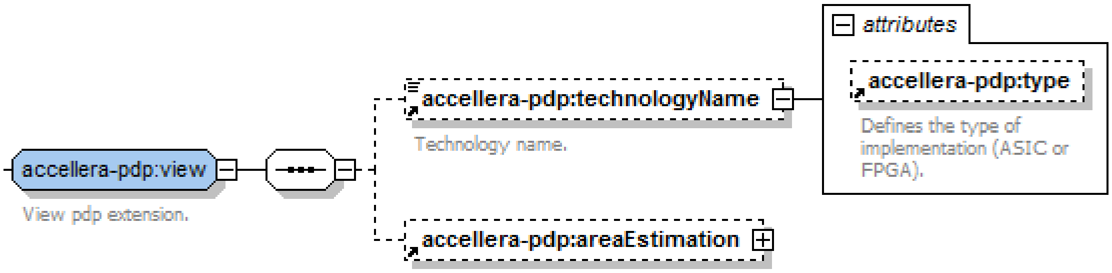
5. Physical design planning extensions

The physical design planning extensions are defined in the “accellera-pdp” namespace. The purpose of physical design planning extensions is to describe vendor extensions meta-data that enables automation of ASIC and FPGA implementation flows.

5.1. View extension

5.1.1. Schema

The following schema defines the contents of the physical design planning view container.


attributes
accellera-pdp:type
Defines the type of implementation (ASIC or FPGA).
accellera-pdp:technologyName
Technology name.
accellera-pdp:view
View pdp extension.
accellera-pdp:areaEstimation

5.1.2. Description

The physical design planning view extension contains the following elements.

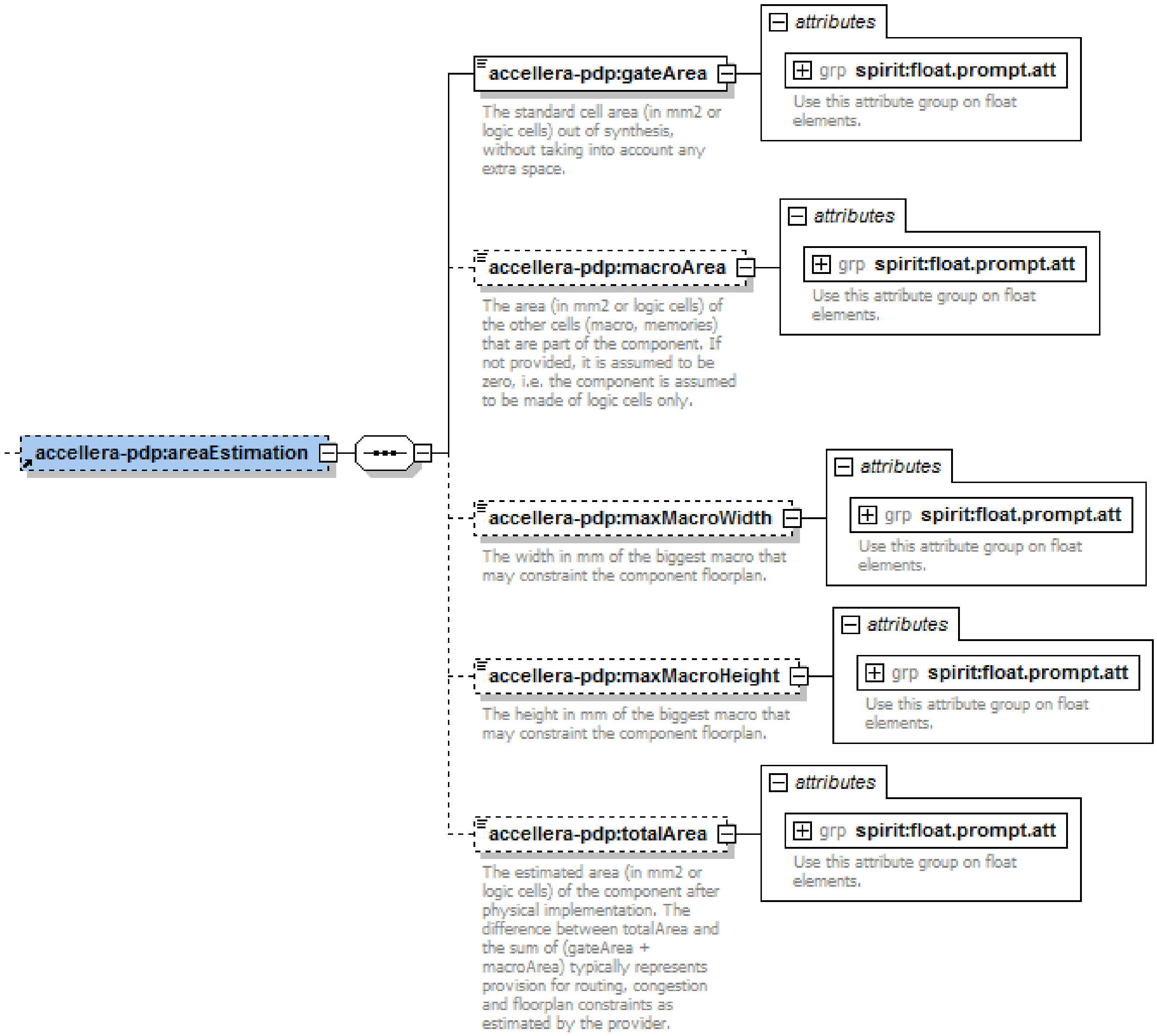
a) **accellera-pdp:technologyName** (optional) describing the name of the technology. This element has a mandatory attribute **accellera-pdp:type** that can take the values ASIC and FPGA indicating the type of technology.

b) **accellera-pdp:areaEstimation (**optional) describing properties of the view in the specified technology. The elements of **areaEstimation** are described in Section 5.1.3

5.1.3. Area estimation

5.1.3.1. Schema

The following schema defines the contents of the area estimation element.


accellera-pdp:areaEstimation
accellera-pdp:gateArea
attributes
grp spirit:float.prompt.att
accellera-pdp:macroArea
attributes
grp spirit:float.prompt.att
accellera-pdp:maxMacroWidth
attributes
grp spirit:float.prompt.att
accellera-pdp:maxMacroHeight
attributes
grp spirit:float.prompt.att
accellera-pdp:totalArea
attributes
grp spirit:float.prompt.att

The standard cell area (in mm2 or logic cells) out of synthesis, without taking into account any extra space.

Use this attribute group on float elements.

The area (in mm2 or logic cells) of the other cells (macro, memories) that are part of the component. If not provided, it is assumed to be zero, i.e. the component is assumed to be made of logic cells only.

Use this attribute group on float elements.

The width in mm of the biggest macro that may constraint the component floorplan.

Use this attribute group on float elements.

The height in mm of the biggest macro that may constraint the component floorplan.

Use this attribute group on float elements.

The estimated area (in mm2 or logic cells) of the component after physical implementation. The difference between totalArea and the sum of (gateArea + macroArea) typically represents provision for routing, congestion and floorplan constraints as estimated by the provider.

Use this attribute group on float elements.

5.1.3.2. Description

The area estimation contains the following elements.

a) **accellera-pdp:gateArea** describing the standard cell area (in mm^2 or number of logic cells depending on the technology type ASIC or FPGA, respectively) out of synthesis without taking into account any extra space.

b) **accellera-pdp:macroArea** (optional) describing area (in mm^2 or number of logic cells depending on the technology type ASIC or FPGA, respectively) of the other cells (macro, memories) that are part of the component. If not present, then the value is assumed to be zero, i.e., the component is assumed to be made of logic cells only.

c) **accellera-pdp:maxMacroWidth** (optional) describing the width in mm of the biggest macro that may constrain the component floor plan.

d) **accellera-pdp:maxMacroHeight** (optional) describing the height in mm of the biggest macro that may constrain the component floor plan.

e) **accellera-pdp:totalArea** (optional) describing the estimated area (in mm^2 or number of logic cells depending on the technology type ASIC or FPGA, respectively) of the component after physical implementation. The difference between **totalArea** and the sum of **gateArea** and **macroArea** typically represents provision for routing, congestion and floor plan constraints as estimated by the provider.

The values may be dependent on one or more parameters to account for configurable IPs.

5.1.4. Example

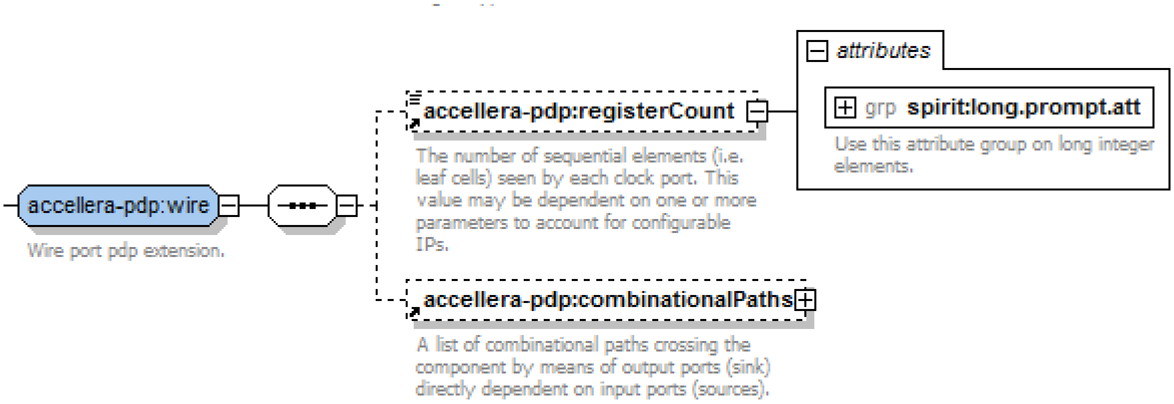
This is an example of a physical design planning view extension.

```
<spirit:view>
  <spirit:name>layout</spirit:name>
  <spirit:envIdentifier>:*Layout:</spirit:envIdentifier>
  <spirit:vendorExtensions>
    <accellera:view>
      <accellera-pdp:technologyName type="asic">cmos032lp
      </accellera-pdp:technologyName>
      <accellera-pdp:areaEstimation>
        <accellera-pdp:maxMacroWidth spirit:resolve="immediate">0.02
        </accellera-pdp:maxMacroWidth>
        <accellera-pdp:gateArea spirit:resolve="dependent"
          spirit:dependency="id(‘param1’)*1.24">1.24
        </accellera-pdp:gateArea>
        <accellera-pdp:macroArea spirit:resolve="dependent"
          spirit:dependency="id(‘param1’)*0.2">0.2
        </accellera-pdp:macroArea>
        <accellera-pdp:maxMacroHeight spirit:resolve="dependent"
          spirit:dependency="id(‘param1’)*0.01">0.01
        </accellera-pdp:maxMacroHeight>
        <accellera-pdp:totalArea spirit:resolve="dependent"
          spirit:dependency="id(‘param1’)*1.44">1.50
        </accellera-pdp:totalArea>
      </accellera-pdp:areaEstimation>
    </accellera:view>
  </spirit:vendorExtensions>
</spirit:view>
```

5.2. Wire extension

5.2.1. Schema

The following schema defines the contents of the physical design planning wire container.


attributes
grp spirit:long.prompt.att
Use this attribute group on long integer elements.
accellera-pdp:registerCount
The number of sequential elements (i.e. leaf cells) seen by each clock port. This value may be dependent on one or more parameters to account for configurable IPs.
accellera-pdp:wire
Wire port pdp extension.
accellera-pdp:combinationalPaths
A list of combinational paths crossing the component by means of output ports (sink) directly dependent on input ports (sources).

5.2.2. Description

The physical design planning wire container has the following elements.

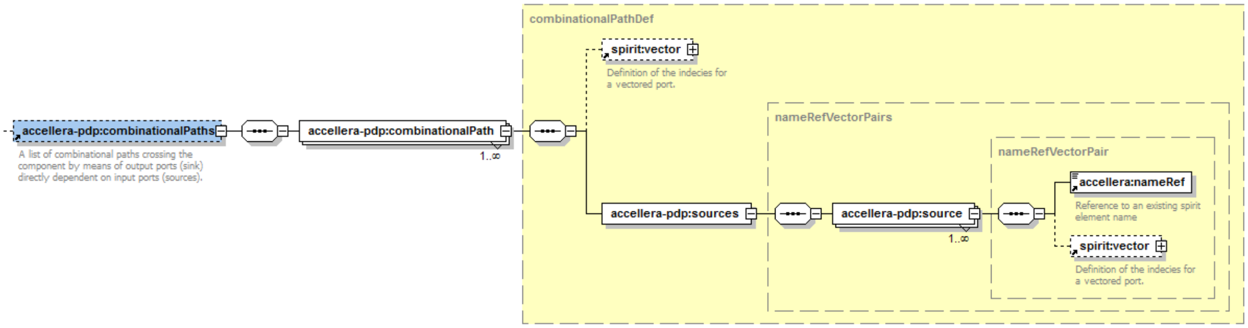
a) **accellera-pdp:registerCount** (optional) specifies the number of sequential elements (i.e. leaf cells) seen by this port in case it is a clock source. This value may be dependent on one or more parameters to account for configurable IPs.

b) **accellera-pdp:combinationalPaths** (optional) is a list of combinational paths crossing the component by means of this output port (sink) directly dependent on input ports (sources); it is defined in Section 5.2.3.

5.2.3. Combinational paths

5.2.3.1. Schema

The following schema defines the contents of the combinational path element.


combinationalPathDef
spirit:vector
Definition of the indecies for a vectored port.
accellera-pdp:combinationalPaths
A list of combinational paths crossing the component by means of output ports (sink) directly dependent on input ports (sources).
accellera-pdp:combinationalPath
1..∞
nameRefVectorPairs
nameRefVectorPair
accellera:nameRef
Reference to an existing spirit element name
accellera-pdp:sources
accellera-pdp:source
1..∞
spirit:vector
Definition of the indecies for a vectored port.

5.2.3.2. Description

The combinational path element contains the following elements.

a) **spirit:vector** (optional) describing the range of the port to which the combinational path applies

b) **accellera-pdp:sources** (mandatory) is a list of **accellera-pdp:source** elements. Each source element has a mandatory **accellera:nameRef** element referring to an input port and an optional **spirit:vector** element indicating the range of that input port.

5.2.4. Example

This is an example of a physical design planning wire extension.

```xml
<spirit:port>
  <spirit:name>myoutput</spirit:name>
  <spirit:wire>
    <spirit:direction>out</spirit:direction>
  </spirit:wire>
  <spirit:vendorExtensions>
    <accellera:wire>
      <accellera-pdp:combinationalPaths>
        <accellera-pdp:combinationalPath>
          <spirit:vector>
            <spirit:left>0</spirit:left>
            <spirit:right>0</spirit:right>
          </spirit:vector>
          <accellera-pdp:sources>
            <accellera-pdp:source>
              <spirit:name>myinput1</spirit:name>
              <spirit:vector>
                <spirit:left>5</spirit:left>
                <spirit:right>5</spirit:right>
              </spirit:vector>
            </accellera-pdp:source>
            <accellera-pdp:source>
              <spirit:name>myinput2</spirit:name>
            </accellera-pdp:source>
          </accellera-pdp:sources>
        </accellera-pdp:combinationalPath>
      </accellera-pdp:combinationalPaths>
    </accellera:wire>
  </spirit:vendorExtensions>
</spirit:port>
```

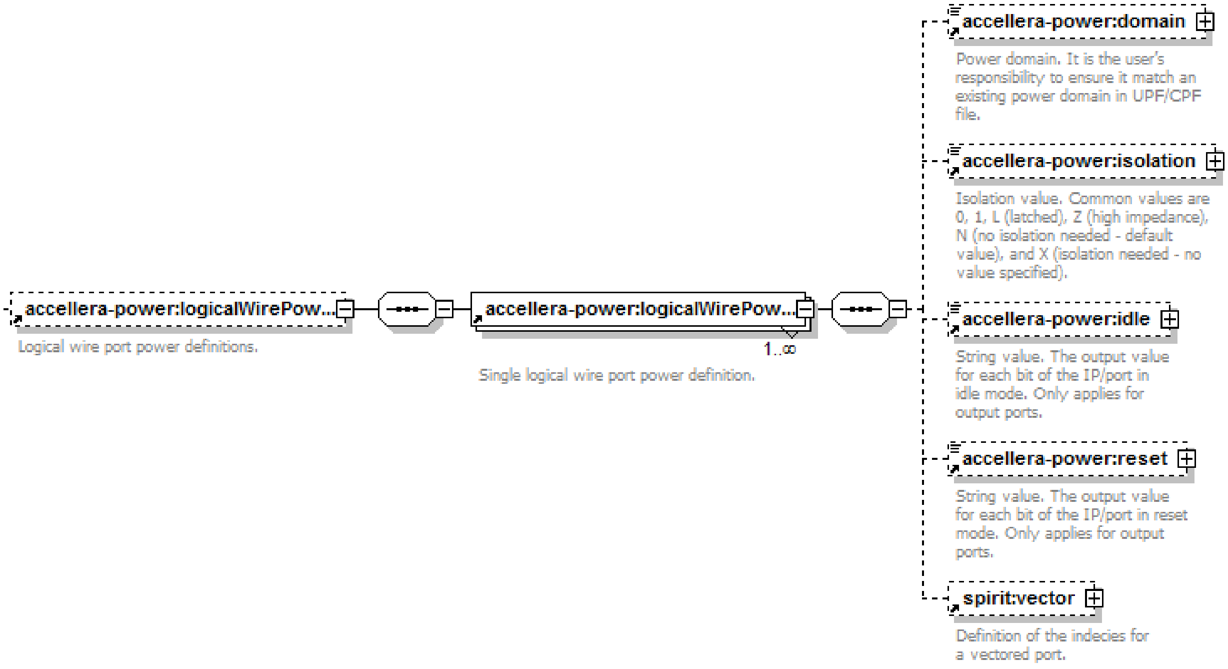
6. Power extensions

The power extensions are defined in the “accellera-power” namespace. The purpose of power extensions is to describe vendor extensions meta-data that enables automation of power flows.

6.1. Logical wire extension

6.1.1. Schema

The following schema defines the content of the power logical wire container.


accellera-power:logicalWirePow...
Logical wire port power definitions.
accellera-power:logicalWirePow...
1..∞
Single logical wire port power definition.
accellera-power:domain
Power domain. It is the user's responsibility to ensure it match an existing power domain in UPF/CPF file.
accellera-power:isolation
Isolation value. Common values are 0, 1, L (latched), Z (high impedance), N (no isolation needed - default value), and X (isolation needed - no value specified).
accellera-power:idle
String value. The output value for each bit of the IP/port in idle mode. Only applies for output ports.
accellera-power:reset
String value. The output value for each bit of the IP/port in reset mode. Only applies for output ports.
spirit:vector
Definition of the indecies for a vectored port.

6.1.2. Description

The logical wire power container contains the following elements.

a) **accellera-power:domain** (optional) specifies the power domain. It is the user’s responsibility to ensure it matches with an existing power domain in an UPF/CPF file.
b) **accellera-power:isolation** (optional) specifies the isolation value. Common values are 0, 1, L (latched), Z (high impedance), N (no isolation needed - default value), and X (isolation needed - no value specified).
c) **accellera-power:idle** (optional) is a string value. The output value for each bit of the port in idle mode. Only applies for output ports.
d) **accellera-power:reset** (optional) is a String value. The output value for each bit of the port in reset mode. Only applies for output ports.
e) **spirit:vector** (optional) is a vector specifying for which port elements this power description applies.

6.1.3. Example

This is an example of a logical wire power extension.

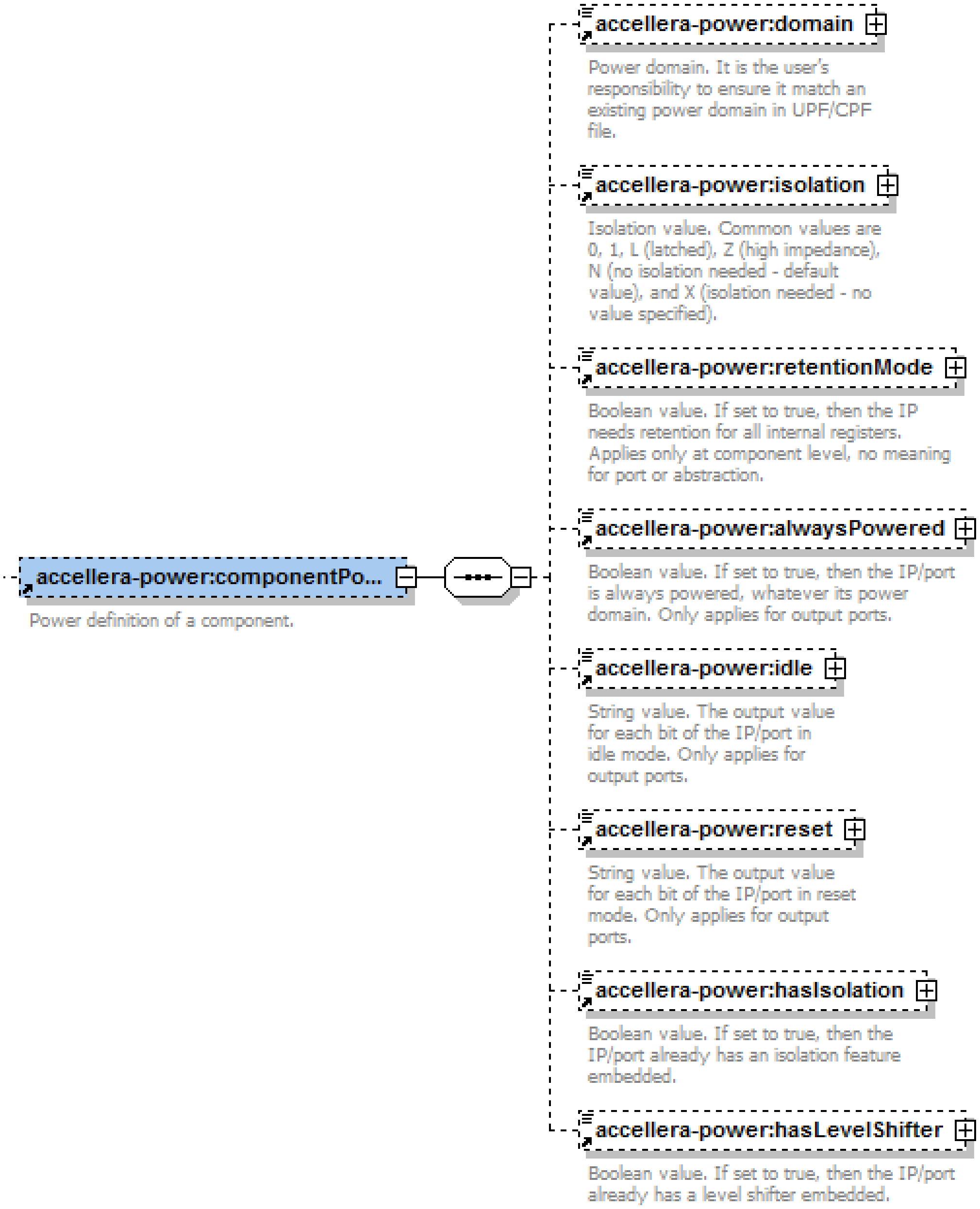
```
<spirit:abstractionDefinition …>
  <spirit:vendor>accellera.org</spirit:vendor>
  <spirit:library>library</spirit:library>
  <spirit:name>absdef</spirit:name>
  <spirit:version>1.0</spirit:version>
  <spirit:busType spirit:vendor="accellera.org" spirit:library="library"
    spirit:name="busdef" spirit:version="1.0"/>
  <spirit:ports>
    <spirit:port>
```

```
      <spirit:logicalName>lo1</spirit:logicalName>
      <spirit:wire/>
    </spirit:port>
    <spirit:port>
      <spirit:logicalName>lo2</spirit:logicalName>
      <spirit:wire/>
      <spirit:vendorExtensions>
        <accellera:logicalWire>
          <accellera-power:logicalWirePowerDefs>
            <accellera-power:logicalWirePowerDef>
              <accellera-power:domain>domain4</accellera-power:domain>
              <accellera-power:isolation>Z</accellera-power:isolation>
              <accellera-power:idle>1</accellera-power:idle>
              <accellera-power:reset>0</accellera-power:reset>
            </accellera-power:logicalWirePowerDef>
          </accellera-power:logicalWirePowerDefs>
        </accellera:logicalWire>
      </spirit:vendorExtensions>
    </spirit:port>
  </spirit:ports>
</spirit:abstractionDefinition>
```

6.2. Component extension

6.2.1. Schema

The following schema defines the content of the power component container.



accellera-power:componentPo...

Power definition of a component.

accellera-power:domain

Power domain. It is the user's responsibility to ensure it match an existing power domain in UPF/CPF file.

accellera-power:isolation

Isolation value. Common values are 0, 1, L (latched), Z (high impedance), N (no isolation needed - default value), and X (isolation needed - no value specified).

accellera-power:retentionMode

Boolean value. If set to true, then the IP needs retention for all internal registers. Applies only at component level, no meaning for port or abstraction.

accellera-power:alwaysPowered

Boolean value. If set to true, then the IP/port is always powered, whatever its power domain. Only applies for output ports.

accellera-power:idle

String value. The output value for each bit of the IP/port in idle mode. Only applies for output ports.

accellera-power:reset

String value. The output value for each bit of the IP/port in reset mode. Only applies for output ports.

accellera-power:hasIsolation

Boolean value. If set to true, then the IP/port already has an isolation feature embedded.

accellera-power:hasLevelShifter

Boolean value. If set to true, then the IP/port already has a level shifter embedded.

6.2.2. Description

The power component container contains the following elements.

a) **accellera-power:domain** (optional) specifies the power domain. It is the user's responsibility to ensure it matches with an existing power domain in an UPF/CPF file.
b) **accellera-power:isolation** (optional) specifies the isolation value. Common values are 0, 1, L (latched), Z (high impedance), N (no isolation needed - default value), and X (isolation needed - no value specified).
c) **accellera-power:retentionMode** (optional) is a boolean value. If set to true, then the component needs retention for all internal registers.
d) **accellera-power:alwaysPowered** (optional) is a boolean value. If set to true, then the component is always powered, whatever its power domain. Only applies for output ports.
e) **accellera-power:idle** (optional) is a string value. The output value for each bit of the component in idle mode. Only applies for output ports.

f) **accellera-power:reset** (optional) is a String value. The output value for each bit of the component in reset mode. Only applies for output ports.
g) **accellera-power:hasIsolation** (optional) is a boolean value. If set to true, then the component already has an isolation feature embedded.
h) **accellera-power:hasLevelShifter** (optional) is a boolean value. If set to true, then the component already has a level shifter embedded.

6.2.3. Example

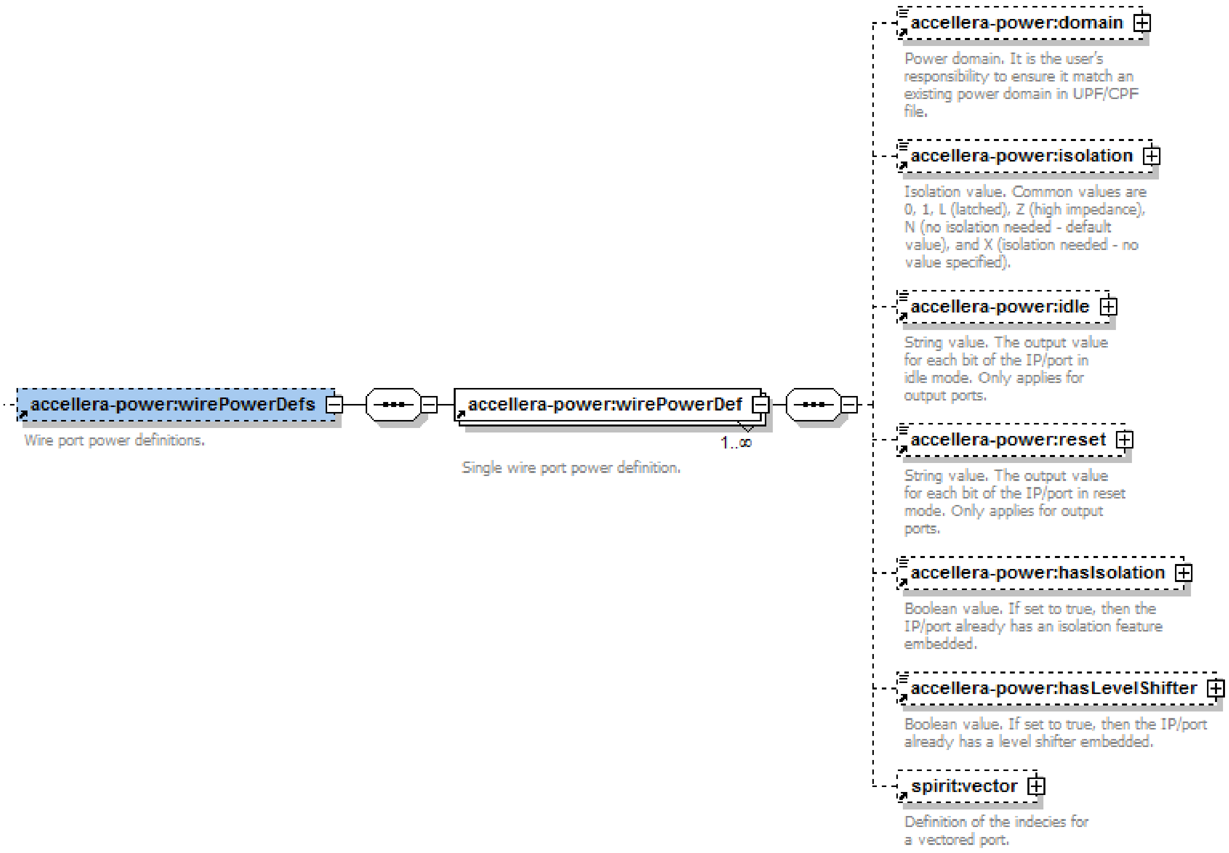
This is an example of a component power extension.

```
<spirit:component …>
  <spirit:vendor>accellera.org</spirit:vendor>
  <spirit:library>library</spirit:library>
  <spirit:name>component</spirit:name>
  <spirit:version>1.0</spirit:version>
  <spirit:vendorExtensions>
    <accellera:component>
      <accellera-power:componentPowerDef>
      <accellera-power:domain>mydomain</accellera-power:domain>
      <accellera-power:isolation>0</accellera-power:isolation>
      </accellera-power:componentPowerDef>
    </accellera:component>
  </spirit:vendorExtensions>
</spirit:component>
```

6.3. Wire extension

6.3.1. Schema

The following schema defines the content of the power wire container.


accellera-power:wirePowerDefs
Wire port power definitions.
accellera-power:wirePowerDef
1..∞
Single wire port power definition.
accellera-power:domain
Power domain. It is the user's responsibility to ensure it match an existing power domain in UPF/CPF file.
accellera-power:isolation
Isolation value. Common values are 0, 1, L (latched), Z (high impedance), N (no isolation needed - default value), and X (isolation needed - no value specified).
accellera-power:idle
String value. The output value for each bit of the IP/port in idle mode. Only applies for output ports.
accellera-power:reset
String value. The output value for each bit of the IP/port in reset mode. Only applies for output ports.
accellera-power:hasIsolation
Boolean value. If set to true, then the IP/port already has an isolation feature embedded.
accellera-power:hasLevelShifter
Boolean value. If set to true, then the IP/port already has a level shifter embedded.
spirit:vector
Definition of the indecies for a vectored port.

6.3.2. Description

The power wire container contains the following elements.

a) **accellera-power:domain** (optional) specifies the power domain. It is the user's responsibility to ensure it matches with an existing power domain in an UPF/CPF file.
b) **accellera-power:isolation** (optional) specifies the isolation value. Common values are 0, 1, L (latched), Z (high impedance), N (no isolation needed - default value), and X (isolation needed - no value specified).
c) **accellera-power:idle** (optional) is a string value. The output value for each bit of the port in idle mode. Only applies for output ports.
d) **accellera-power:reset** (optional) is a String value. The output value for each bit of the port in reset mode. Only applies for output ports.
e) **accellera-power:hasIsolation** (optional) is a boolean value. If set to true, then the port already has an isolation feature embedded.
f) **accellera-power:hasLevelShifter** (optional) is a boolean value. If set to true, then the port already has a level shifter embedded.
g) **spirit:vector** (optional) is a vector specifying for which port elements this power description applies.

6.3.3. Example

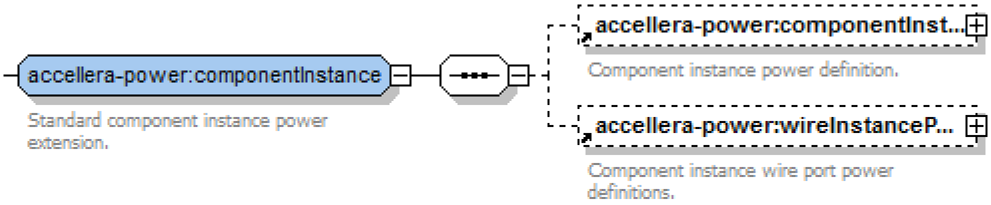
This is an example of a wire power extension.

```
<spirit:port>
  <spirit:name>pc</spirit:name>
  <spirit:wire>
    <spirit:direction>in</spirit:direction>
    <spirit:vector>
      <spirit:left>7</spirit:left>
      <spirit:right>0</spirit:right>
    </spirit:vector>
  </spirit:wire>
  <spirit:vendorExtensions>
    <accellera:wire>
      <accellera-power:wirePowerDefs>
        <accellera-power:wirePowerDef>
          <accellera-power:domain>domain2</accellera-power:domain>
          <accellera-power:isolation>L</accellera-power:isolation>
        </accellera-power:wirePowerDef>
        <accellera-power:wirePowerDef>
          <accellera-power:domain>domain3</accellera-power:domain>
          <spirit:vector>
            <spirit:left>3</spirit:left>
            <spirit:right>0</spirit:right>
          </spirit:vector>
        </accellera-power:wirePowerDef>
      </accellera-power:wirePowerDefs>
    </accellera:wire>
  </spirit:vendorExtensions>
</spirit:port>
```

6.4. Component instance extension

6.4.1. Schema

The following schema defines the content of the power component instance container.


accellera-power:componentInstance
Standard component instance power extension.
accellera-power:componentInst...
Component instance power definition.
accellera-power:wireInstanceP...
Component instance wire port power definitions.

6.4.2. Description

The power component instance container contains the following elements.

a) **accellera-power:componentInstancePowerDef** (optional) specifies the power information for component instances. This element is a component power definition as defined in Section 6.2.

b) **accellera-power:wireInstancePowerDefs** (optional) the power information for component instances. This element is a wire power definition as defined in Section 6.4.3.1.

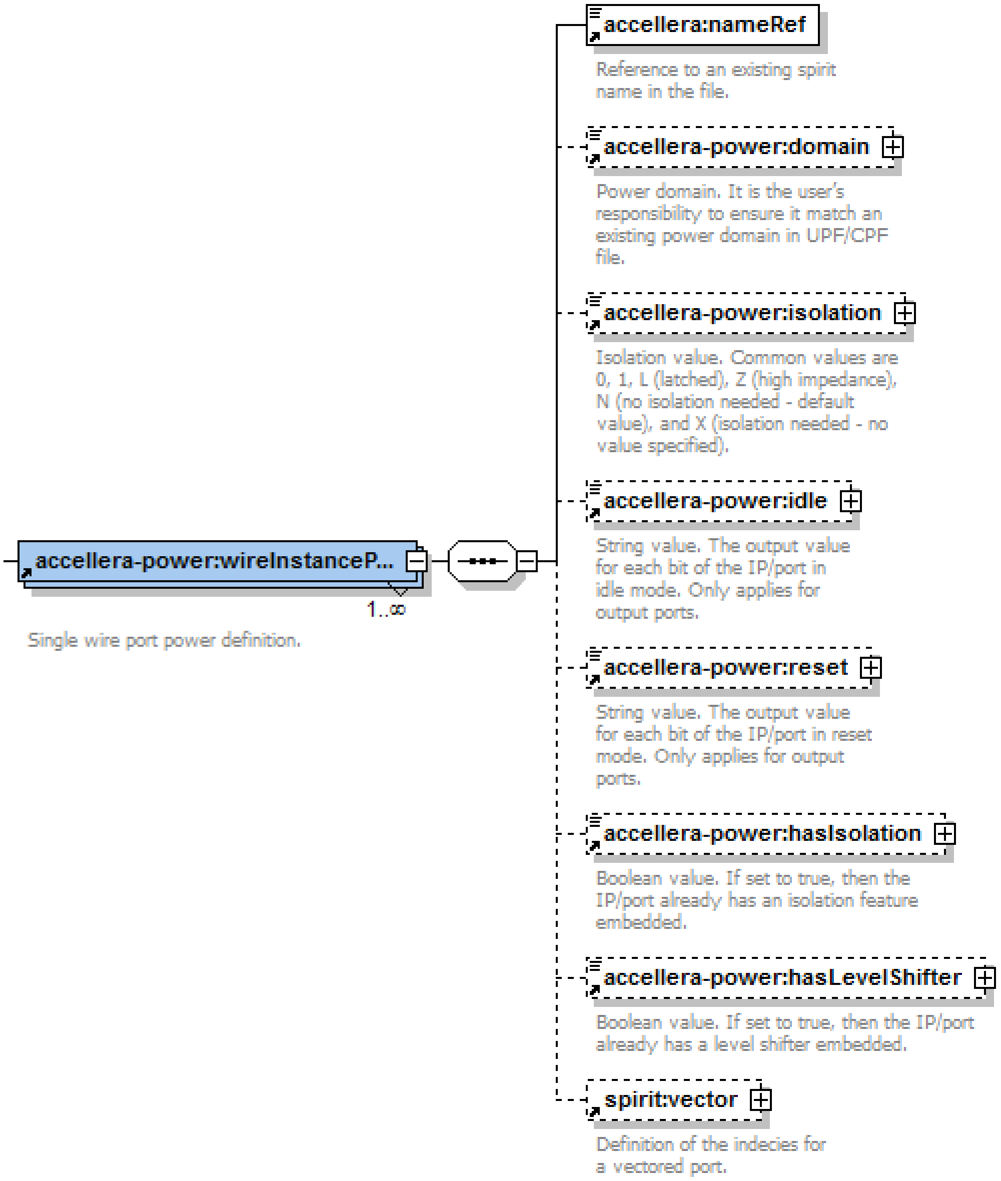
6.4.3. Example

This is an example of a component instance power extension.

```
<spirit:componentInstance>
  <spirit:instanceName>myip1</spirit:instanceName>
  <spirit:componentRef spirit:vendor="Mds" spirit:library="test" spirit:name="comp"
    spirit:version="1.0"/>
  <spirit:vendorExtensions>
    <accellera:componentInstance>
      <accellera-power:componentInstancePowerDef>
        <accellera-power:domain>domain6</accellera-power:domain>
        <accellera-power:isolation>L</accellera-power:isolation>
        <accellera-power:idle>1</accellera-power:idle>
        <accellera-power:reset>1</accellera-power:reset>
      </accellera-power:componentInstancePowerDef>
    </accellera:componentInstance>
  </spirit:vendorExtensions>
</spirit:componentInstance>
```

6.4.3.1. Wire instance power definition

The following schema defines the content of the wire instance power definition.


accellera-power:wireInstanceP...
1..∞
Single wire port power definition.
accellera:nameRef
Reference to an existing spirit name in the file.
accellera-power:domain
Power domain. It is the user's responsibility to ensure it match an existing power domain in UPF/CPF file.
accellera-power:isolation
Isolation value. Common values are 0, 1, L (latched), Z (high impedance), N (no isolation needed - default value), and X (isolation needed - no value specified).
accellera-power:idle
String value. The output value for each bit of the IP/port in idle mode. Only applies for output ports.
accellera-power:reset
String value. The output value for each bit of the IP/port in reset mode. Only applies for output ports.
accellera-power:hasIsolation
Boolean value. If set to true, then the IP/port already has an isolation feature embedded.
accellera-power:hasLevelShifter
Boolean value. If set to true, then the IP/port already has a level shifter embedded.
spirit:vector
Definition of the indecies for a vectored port.

6.4.3.2. Description

The power wire instance port power definition contains an **accellera:nameRef** element specifying the name of component port for which this wire instance port power definition applies. All other elements are identical to the elements in a wire port power definition described in Section 6.3.

6.4.3.3. Example

This is an example of a wire instance power extension within a component instance power extension.

```
<spirit:componentInstance>
  <spirit:instanceName>myip1</spirit:instanceName>
  <spirit:componentRef spirit:vendor="Mds" spirit:library="test" spirit:name="comp"
    spirit:version="1.0"/>
  <spirit:vendorExtensions>
    <accellera:componentInstance>
```

```
      <accellera-power:wireInstancePowerDefs>
        <accellera-power:wireInstancePowerDef>
          <accellera:nameRef>pd</accellera:nameRef>
          <accellera-power:domain>domain7</accellera-power:domain>
          <accellera-power:isolation>X</accellera-power:isolation>
        </accellera-power:wireInstancePowerDef>
      </accellera-power:wireInstancePowerDefs>
    </accellera:componentInstance>
  </spirit:vendorExtensions>
</spirit:componentInstance>
```

7. Semantic Consistency Rules

Accellera vendor extensions need to obey to the following semantic consistency rules in order to be valid.

7.1. Core Rules

Table 5: Core semantic consistency rules.

Rule Number	Rule	Single doc check	Notes
SCR CORE.1	The vector of a portParameter must be a sub-vector of the port.	yes	See also 3.1.3.
SCR CORE.2	The vectors of two portParameters of the same port that have the same portParameter names cannot overlap.	yes	See also 3.1.3.
SCR CORE.3	A port with direction out shall not have a driver extension	yes	See also 3.2.3.
SCR CORE.4	The number of values in a defaultValue must be equal to the number of port elements.	yes	See also 3.2.3.

7.2. Analog and Mixed Signal Rules

Table 6: Analog and mixed signal semantic consistency rules.

Rule Number	Rule	Single doc check	Notes
SCR AMS.1	A component port with direction out and an analog and mixed signal extension does not count in a port connection equivalence class, i.e., multiple drivers are allowed on analog and mixed signals.	no	See also SCR 6.9.

7.3. Physical Design Planning Rules

Table 7: Physical design planning semantic consistency rules.

Rule Number	Rule	Single doc check	Notes
SCR PDP.1	totalArea should be at least equal to the sum of gateArea and macoArea	yes	See also 5.1.3.
SCR PDP.2	A view containing areaEstimation must have a technologyName with a type attribute value .	yes	See also 5.1.
SCR PDP.3	A view containing areaEstimation must have an envIdentifier containing “Layout”	yes	See also 5.1.
SCR PDP.4	if a view refers to a file set with a LEF or XDC files, then it cannot contain an areaEstimation.	yes	See also 5.1.
SCR PDP.5	Only input ports can have registerCounts.	yes	See also 5.2.
SCR PDP.6	A port with a registerCount must be mapped in an interface onto a logical port with the isClock qualifier.	yes	See also 5.2.
SCR PDP.7	In a combinationalPath, both sink and source must be a single bit.	yes	See also 5.2.
SCR PDP.8	Files in file sets referenced from a view containing a technologyName with a type equal to “ASIC” or “FPGA” must be of userFileType “LEF” or “XDC”, respectively.	yes	See also 5.1.

7.4. Power Rules

Table 8: Power semantic consistency rules.

Rule Number	Rule	Single doc check	Notes
SCR PWR.1	The vector of a logical or component port power extension must be a sub-vector of the port.	yes	See also 6.1and 6.3.
SCR PWR.2	The vectors of two logical or component port power extensions of the same port cannot overlap.	yes	See also 6.1and 6.3.
SCR PWR.3	Only a logical or component port with direction out can contain an idle element.	yes	See also 6.1and 6.3.
SCR PWR.4	Only a logical or component port with direction out can contain a reset element.	yes	See also 6.1and 6.3.
SCR PWR.5	Power extensions have priorities. Component instance extensions have priority over component port extensions. Component port extensions have priority over component extensions. Component extensions have priority over logical port extensions.	no	See also 6.1, 6.2, 6.3, and 6.4.
SCR PWR.6	The value of a domain element in a power extension must match with a power domain in a UPF/CPF file.	Requires external knowledge	See also 6.1, 6.2, 6.3, and 6.4.