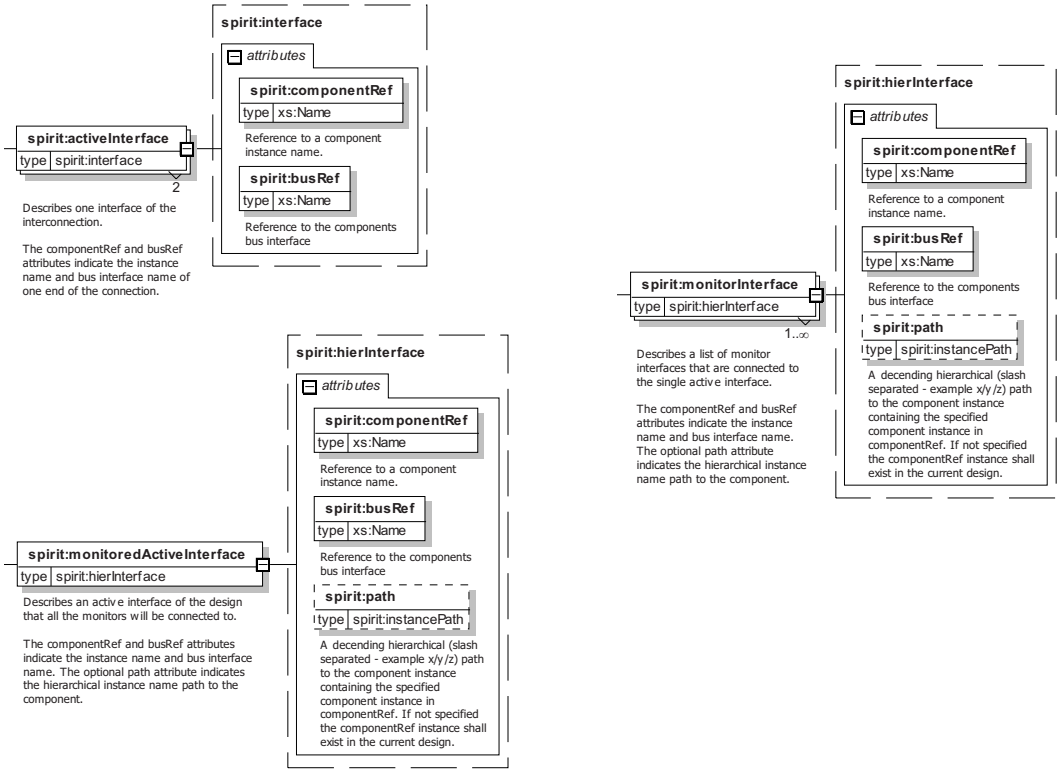


7.4.2 Description

The **activeInterface**, **monitoredInterface**, or **monitorInterface** element specifies the bus interface of a design component instance that is part of an interconnection or a monitor interconnection. They all have the following attributes.

a) **componentRef** (mandatory) references the instance name of a component present in the design if the path attribute is not present. This component instance name needs to exist in the specified design. The **componentRef** attribute is of type ***Name***. See 6.1.

b) **busRef** (mandatory) references one of the component bus interfaces. This specific bus interface needs to exist on the specified component instance. The **busRef** attribute is of type ***Name***. See 6.5.

The **monitoredActiveInterface** and **monitorInterface** elements have the following attribute.

path (optional) defines the hierarchical path of instance names to the design that contains the component instance specified in the **componentRef** attribute. The path is a slash (/) separated list of instance names. If the **path** attribute is not present, the component referenced by **componentRef** needs to exist in the current design. The **path** attribute is of type ***instancePath***. See D.5.

See also: SCR 2.1, SCR 2.16, SCR 4.1, and SCR 4.2.

7.4.3 Example

The following example shows a monitored interface referring to the `ambaAPB` bus interface on the component instance `i_timers` in the design within the component with instance name `apbsubsys/group1` and a monitor interface referring to the `ambaAPBMonitor` bus interface on the monitor instance `i_monitor` in the design within the component with instance name `umon`.

```
<spirit:monitoredInterface spirit:path="apbsubsys/group1"
   spirit:componentRef="i_timers" spirit:busRef="ambaAPB"/>
<spirit:monitorInterface spirit:path="umon" spirit:componentRef="i_monitor"
   spirit:busRef="ambaAPBMonitor"/>
```

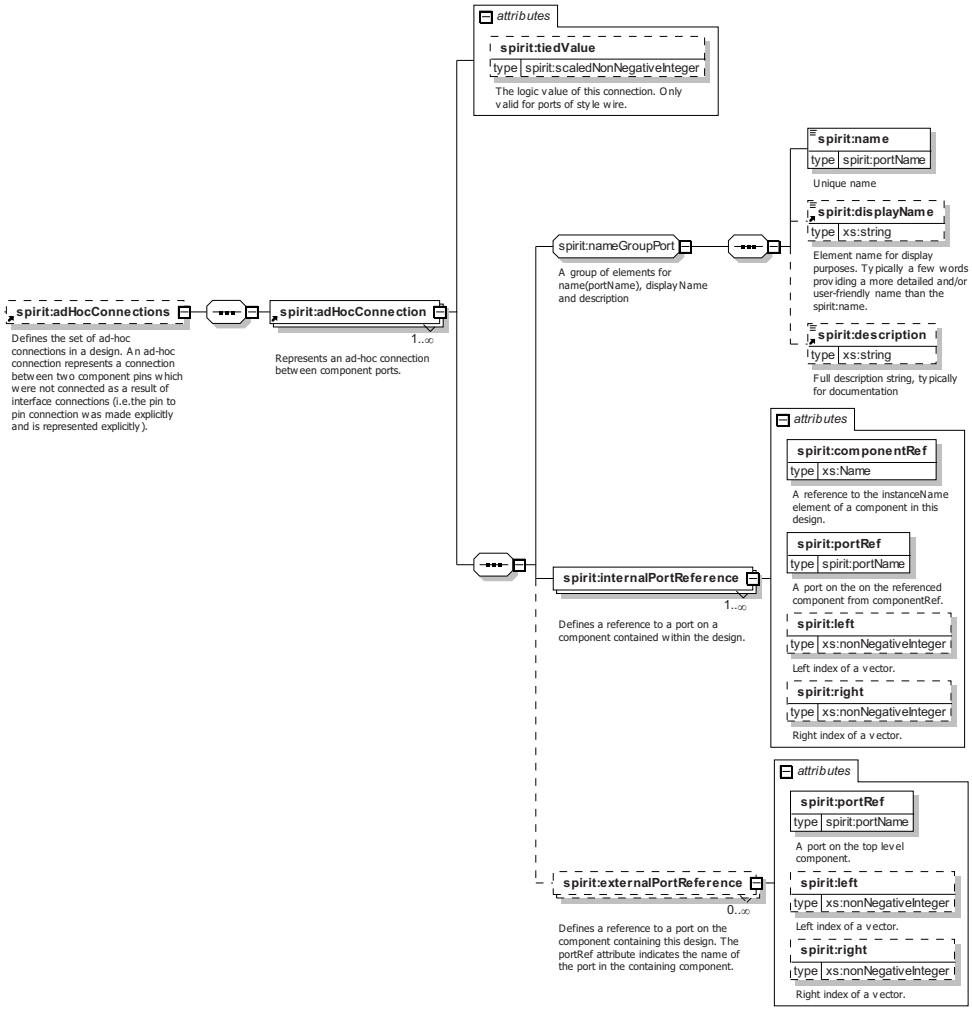
7.5 Design ad hoc connections

The name *ad hoc* is used for connections that are made on a port-by-port basis and not done through the higher-level bus interface. The same **ports** that make up a **busInterface** can be used in ad hoc connections.

IP-XACT supports two cases of ad hoc connections: the wire connection (between ports having a wire style) and the transactional connection (between ports having a transactional style). The direct connection between a wire-style port and a transactional-style port is not allowed; a specific adapter component needs to be inserted in between them.

7.5.1 Schema

The following schema details the information contained in the **adHocConnections** element, which may appear as an element inside the top-level **design** element.



7.5.2 Description

The **adHocConnections** element contains an unbounded list of **adHocConnection** elements. An **adHocConnection** specifies connections between component instance ports or between component instance ports and ports of the encompassing component (in the case of a hierarchical component). Each **adHocConnection** element has a **tiedValue** (optional) attribute that specifies a fixed logic (`1` and `0`) value for this connection. The **tiedValue** attribute is of type ***scaledNonNegativeInteger***. The **adHocConnection** element contains the following subelements.

a) ***nameGroup*** group is defined in C.1. The **name** elements shall be unique within the containing **adHocConnections** element.

b) **internalPortReference** (mandatory) references the port of a component instance. This element has four attributes.

   1) **componentRef** (mandatory) references the component instance name for the port. The **componentRef** attribute is of type ***Name***. See 6.1.

   2) **portRef** (mandatory) references the port name on the specific component instance. The **portRef** attribute is of type ***Name***. See 6.11.3.

   3) **left** and **right** (optional) specify a portion of the port range. The **left** and **right** attributes are of type ***nonNegativeInteger***.

c) **externalPortReference** (optional) references a port of the encompassing component where this design is referred (for hierarchical ad hoc connections). This element has three attributes.

   1) **portRef** (mandatory) references the port name on the encompassing component. The **portRef** attribute is of type ***Name***. See 6.11.3.

   2) **left** and **right** (optional) specify a portion of the port range. The **left** and **right** attribute is of type ***nonNegativeInteger***.

See also: SCR 6.14.

7.5.3 Example

The following example shows two ad hoc connections. The first one, `d1e1074`, connects port `irlin` on component instance `i_irqctrl` and port `irqvec` on component instance `i_leon2Proc`. The second one, `i_leon2Proc_mresult`, connects port `mresult` on component instance `i_leon2Proc` and port `i_leon2Proc_mresult` of the encompassing component.

```
<spirit:adHocConnections>
   <spirit:adHocConnection>
      <spirit:name>d1e1074</spirit:name>
      <spirit:internalPortReference spirit:componentRef="i_irqctrl"
spirit:portRef="irlin" spirit:left="3"
         spirit:right="0"/>
      <spirit:internalPortReference spirit:componentRef="i_leon2Proc"
spirit:portRef="irqvec"
         spirit:left="3" spirit:right="0"/>
   </spirit:adHocConnection>
   <spirit:adHocConnection>
      <spirit:name>i_leon2Proc_mresult</spirit:name>
      <spirit:internalPortReference spirit:componentRef="i_leon2Proc"
spirit:portRef="mresult"
         spirit:left="31" spirit:right="0"/>
      <spirit:externalPortReference spirit:portRef="i_leon2Proc_mresult"/>
   </spirit:adHocConnection>
</spirit:adHocConnections>
```

7.5.4 Ad hoc wire connection

For ad hoc connections between wire-style ports, IP-XACT requires:

— The style of each port be the same style (i.e., **wire**).

— The bits of the ports are connected from left to right. In the **internalPortReference** element, **left** and **right** define the actual bits to connect.

See also: SCR 6.9 and SCR 6.27.

Example

This is an example of these rules being applied.

```
<spirit:adHocConnection>
   </spirit:internalPortReference componentRef="U1" portRef="A"
      left="8" right="1">
   </spirit:internalPortReferencenal componentRef="U2" portRef="B"
      left="7" right="0">
</spirit:adHocConnection>
```

Implies these connections:

```
U1/A[8] = U2/B[7]
U1/A[7] = U2/B[6]
U1/A[6] = U2/B[5]
U1/A[5] = U2/B[4]
U1/A[4] = U2/B[3]
U1/A[3] = U2/B[2]
U1/A[2] = U2/B[1]
U1/A[1] = U2/B[0]
```

NOTE—The **typeName**s do not have to match between the two ports, it is up to the DE or simulator to potentially resolve unmatching types, e.g., it is possible to connect a VHDL `std_logic` port to a SystemC `sc_logic` port.

7.5.5 Ad hoc transactional connection

For ad hoc transactional connections, IP-XACT requires:

— The style of each port be the same style (i.e., **transactional**).

— If defined, the **transTypeDef**/**typeName** name of each port are the same (e.g., `sc_tlm_port`).

— The **service**/**serviceTypeDef**/**typeName**s match.

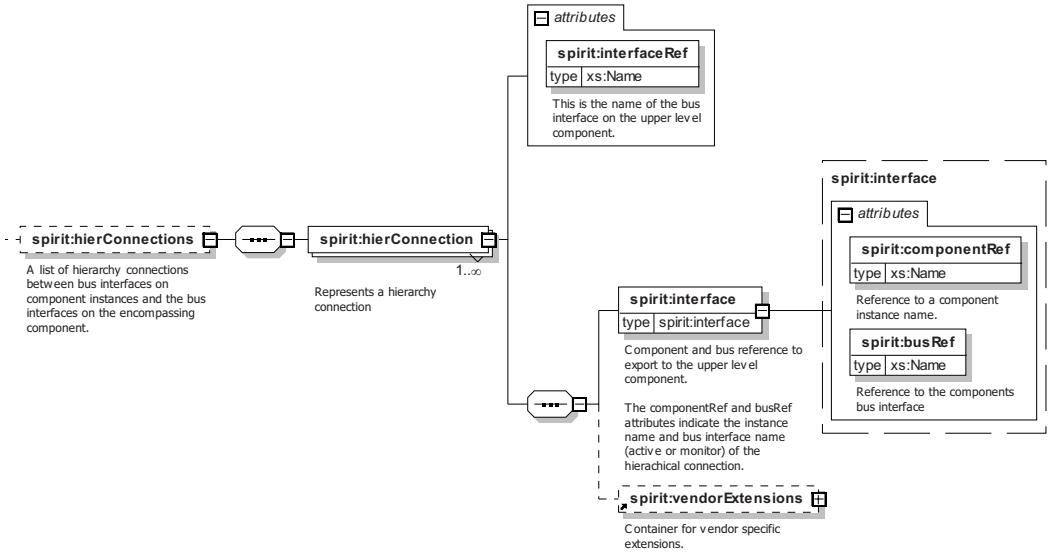
Also, two ports with a **requires** initiative can be connected. This means they would both connect to a mediated link (e.g., a wire, buffer, FIFO, or any complex link) in a top SystemC or SystemVerilog netlist. This mediated link provides the protocol interfaces required by each port. The name, type, and parameters of this mediated link are not defined by IP-XACT, but could be given as input to a netlister generator.

See also: SCR 6.10.

7.6 Design hierarchical connections

7.6.1 Schema

The following schema details the information contained in the **hierConnections** element, which may appear as an element inside the top-level **design** element.



7.6.2 Description

The **hierConnections** element contains an unbounded list of **hierConnection** elements. **hierConnection** represents a hierarchical interface connection between a bus interface on the encompassing component and a bus interface on a component instance of the design. **hierConnection** contains an **interfaceRef** (mandatory) attribute that provides one end of the interconnection; it is the name of the bus interface on the encompassing component (see 6.5.1). The **interfaceRef** attribute is of type ***Name***. The name of the ports and the mapping to this interface are defined in the referencing hierarchical component. The **hierConnection** element contains the following elements and attributes.

a) **interface** (mandatory) specifies the component instance bus interface for connection to the encompassing component; only one **interface** is allowed. The **interface** element may reference an active interface or a monitor interface. The **interface** element is of type ***interface***, see 7.4.

b) **vendorExtensions** (optional) adds any extra vendor-specific data related to the hierarchical interface connection. See C.10.

See also: SCRs in Table B.10 and Table B.11.

7.6.3 Example

The following example shows a hierarchical interconnection between the `AHBReset_1` bus interface on the encompassing component and the `AHBReset` bus interface on the `i_ahbbus` component instance.

```
<spirit:hierConnections>
   <spirit:hierConnection spirit:interfaceRef="AHBReset_1">
      <spirit:activeInterface spirit:componentRef="i_ahbbus"
spirit:busRef="AHBReset"/>
   </spirit:hierConnection>
</spirit:hierConnections>
```

8. Abstractor descriptions

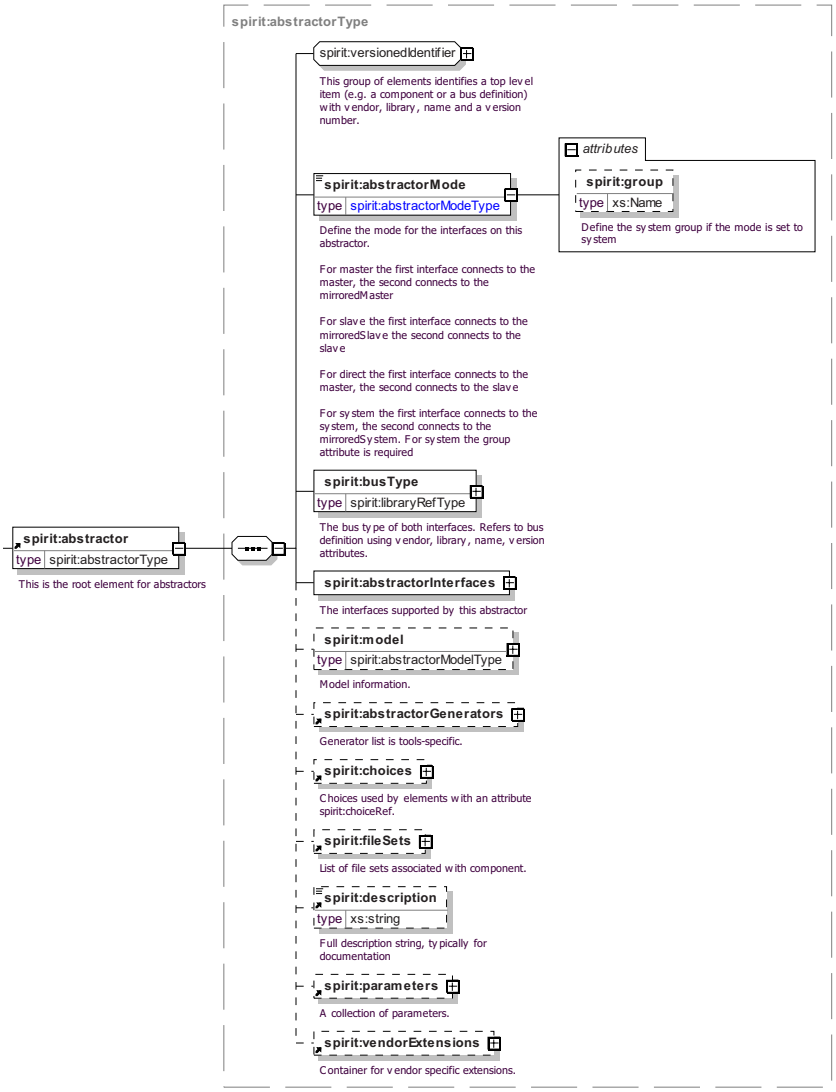
Designs that incorporate IP models using different interface modeling styles (e.g., TLM and RTL modeling styles) may contain interconnections between such component interfaces using different abstractions of the same bus type. An IP-XACT description may describe how such interconnections are to be made using a special-purpose object called an *abstractor*. An abstractor is used to connect between two different abstractions of the same bus type (e.g., an APB_RTL and an APB_TLM). An abstractor shall only contain two interfaces, which shall be of the same bus definition and different abstraction definitions.

Unlike a component, an abstractor is not referenced from a design description, but instead is referenced from a design configuration description. See Clause 10.

8.1 Abstractor

8.1.1 Schema

The following schema details the information contained in the **abstractor** element, which is one of the seven top-level elements in the IP-XACT specification used to describe an abstractor.



8.1.2 Description

Each element of an **abstractor** is detailed in the rest of this clause; the main sections of an **abstractor** are:

a) ***versionedIdentifier*** group provides a unique identifier, made up of four subelements for a top-level IP-XACT element. See C.6.

b) **abstractorMode** (mandatory) determines the mode of the two interfaces contained in **abstractorInterfaces**. The abstractor can be inserted in a connection between two instances or between an instance and an exported interface. The **abstractorMode** element can take one of the following four values.

1) **master** specifies for

i) master to mirrored-master connection—the first interface connects to the master interface, the second connects to the mirrored-master interface;

ii) exported master connection—the first interface connects to the master interface, the second connects to the exported interface;

iii) exported mirrored-master connection—the first interface connects to the exported interface, the second connects to the mirrored-master interface.

2) **slave** specifies for

i) mirrored-slave to slave connection—the first interface connects to the mirrored-slave interface, the second connects to the slave interface;

ii) exported slave connection—the first interface connects to the exported interface, the second connects to the slave interface;

iii) exported mirrored-slave connection—the first interface connects to the mirrored-slave interface, the second connects to the exported interface.

3) **direct** specifies the first interface connects to the master interface, the second connects to the slave interface. This option is not allowed for an exported interface.

4) **system** specifies for

i) system to mirrored-system connection—the first interface connects to the system interface, the second connects to the mirrored-system interface;

ii) exported system connection—the first interface connects to the system interface, the second connects to the exported interface;

iii) exported mirrored-system connection—the first interface connects to the exported interface, the second connects to the mirrored-system interface.

The **group** (mandatory, when **abstractorMode="system"**) attribute defines the name of the group to which this system interface belongs. This attribute is of type ***Name***, which indicates the value of this group shall be unique inside the **abstractor** element. The specified value of **group** needs to be a group defined in the referenced abstraction definition. A connection between a **system** and **mirroredSystem** interfaces shall have matching group names.

c) **busType** (mandatory) specifies the bus definition this bus interface references. A bus definition (see 5.2) describes the high-level attributes of a bus description. The **busType** element is of type ***libraryRefType*** (see C.7); it contains four attributes to specify the referenced VLNV.

d) **abstractorInterfaces** (mandatory) are interfaces having the same bus type, but differing abstraction types. See 8.2.

e) **model** (optional) specifies all the different views, ports, and model configuration parameters of the abstractor. See 8.3.

f) **abstractorGenerators** (optional) specifies a list of generator programs attached to this abstractor. See 8.7.

g) **choices** (optional) specifies multiple enumerated lists, which are referenced by other sections of this abstractor description. See 6.14.

h) **fileSets** (optional) specifies groups of files and possibly their function for reference by other sections of this abstractor description. See 6.13.

i) **description** (optional) allows a textual description of the abstractor. The **description** element is of type ***string***.

j) **parameters** (optional) describes any **parameter** that can be used to configure or hold information related to this abstractor. See C.11.

k) **vendorExtensions** (optional) contains any extra vendor-specific data related to the abstractor. See C.10.

See also: SCR 1.9, SCR 1.10, and SCR 3.16.

8.1.3 Example

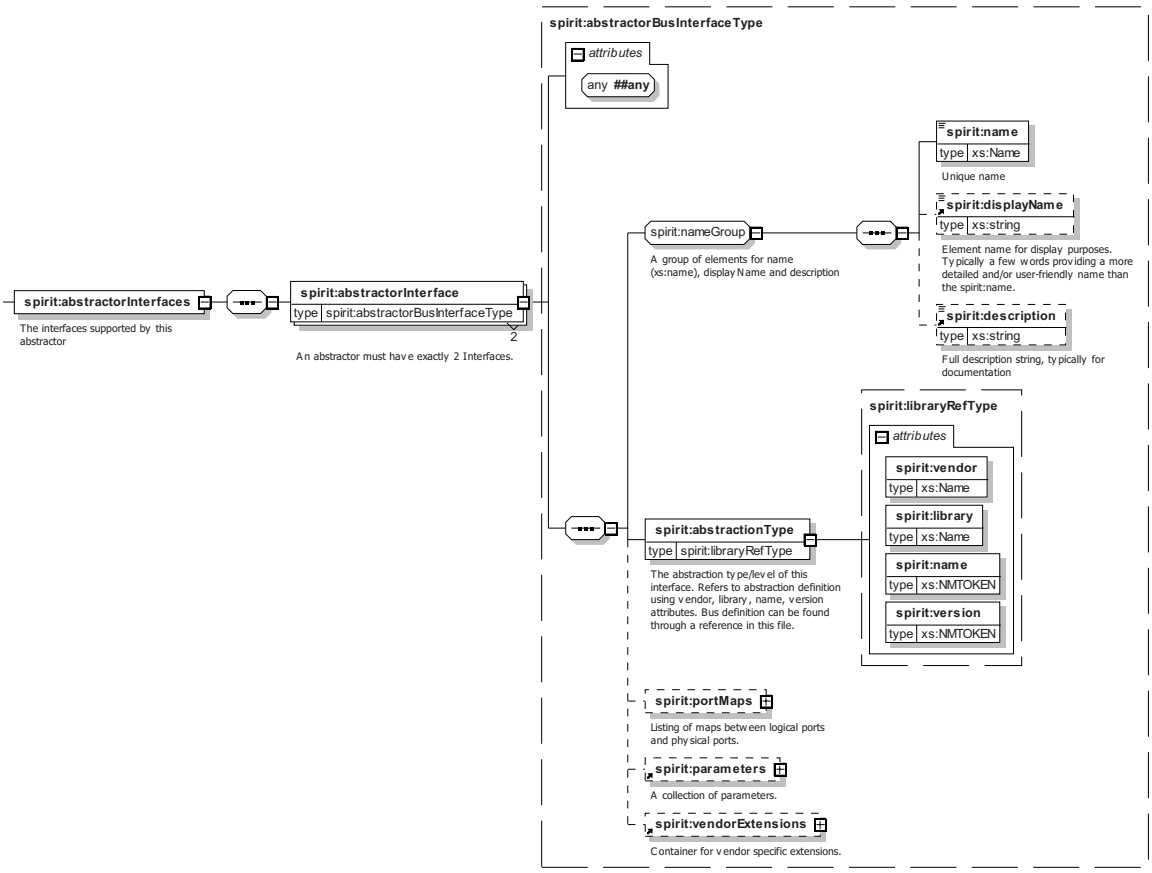
The following example shows a simple slave abstractor having `AHB UT` and `AHB LT` interfaces.

```
<spirit:abstractor>
   <spirit:vendor>spiritconsortium.org</spirit:vendor>
   <spirit:library>Leon2</spirit:library>
   <spirit:name>pv2rtl</spirit:name>
   <spirit:version>1.5</spirit:version>
   <spirit:abstractorMode>slave</spirit:abstractorMode>
   <spirit:busType spirit:vendor="amba.com" spirit:library="AMBA2"
   spirit:name="AHB" spirit:version="r2p0_5"/>
   <spirit:abstractorInterfaces>
      <spirit:abstractorInterface>
         <spirit:name>UTinterface</spirit:name>
         <spirit:abstractionType
            spirit:vendor="spiritconsortium.org"
            spirit:library="Leon2"
            spirit:name="AHB_UT"
            spirit:version="1.0"/>
      </spirit:abstractorInterface>
      <spirit:abstractorInterface>
         <spirit:name>LTinterface</spirit:name>
         <spirit:abstractionType
            spirit:vendor="spiritconsortium.org"
            spirit:library="Leon2"
            spirit:name="AHB_LT"
            spirit:version="1.0"/>
      </spirit:abstractorInterface>
   </spirit:abstractorInterfaces>
</spirit:abstractor>
```

8.2 Abstractor interfaces

8.2.1 Schema

The following schema defines the information contained in the **abstractorInterfaces** element, which appears within an **abstractor** description.



8.2.2 Description

The **abstractorInterfaces** element contains a list of two **abstractorInterface** elements. Each **abstractorInterface** element defines properties of this specific interface in an abstractor. The **abstractorInterface** element also allows for vendor attributes to be applied. Each **abstractorInterface** contains the following elements.

a) ***nameGroup*** group is defined in C.1. The **name** elements shall be unique within the containing **abstractor** element.

b) **abstractionType** (mandatory) specifies the abstraction definition where this bus interface is referenced. An abstraction definition describes the low-level attributes of a bus description (see 5.3). The **abstractionType** element is of type ***libraryRefType*** (see C.7); it contains four attributes to specify the referenced VLNV.

c) **portMaps** (optional) describes the mapping between the abstraction definition's logical ports and the abstractor's physical ports. See 6.5.6.

d) **parameters** (optional) specifies any parameter data value(s) for this bus interface. See C.11.

e) **vendorExtensions** (optional) holds any vendor-specific data from other namespaces, which is applicable to this bus interface. See C.10.

8.2.3 Example

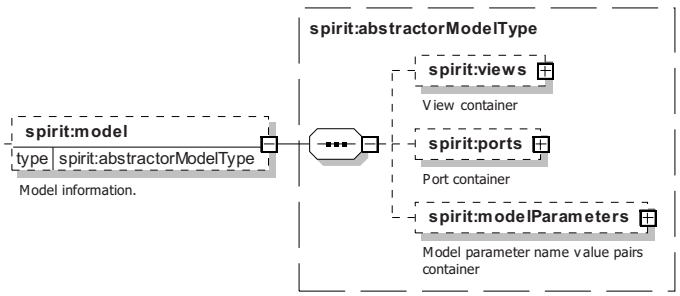
This example shows an **abstractorInterface** of type `AHB_PV`, which includes a single `portMap` between the logical port `PV_TRANS` and the abstractor physical port `ahb_slave_port`.

```
<spirit:abstractorInterface>
   <spirit:name>PVinterface</spirit:name>
   <spirit:abstractionType
      spirit:vendor="spiritconsortium.org"
      spirit:library="Leon2"
      spirit:name="AHB_PV"
      spirit:version="1.0"/>
   <spirit:portMaps>
      <spirit:portMap>
         <spirit:logicalPort>
            <spirit:name>PV_TRANS</spirit:name>
         </spirit:logicalPort>
         <spirit:physicalPort>
            <spirit:name>ahb_slave_port</spirit:name>
         </spirit:physicalPort>
      </spirit:portMap>
   </spirit:portMaps>
</spirit:abstractorInterface>
```

8.3 Abstractor models

8.3.1 Schema

The following schema defines the information contained in the abstractor **model** element, which may appear within an **abstractor** description.



8.3.2 Description

The **model** element describes the views, ports, and model related parameters of an abstractor. A **model** element may contain the following.

a) **views** (optional) contains a list of all the views for this object. An object may have many different views. An RTL view may describe the source hardware module/entity with its pin interface; a software view may define the source device driver C file with its `.h` interface; a documentation view may define the written specification of this IP. See 8.4.

b) **ports** (optional) contains the list of ports for this object. A ports is an external connection from the object. An object may only have one set of ports that shall be valid for all views. See 8.5.

c) **modelParameters** (optional) contains a list of parameters that are needed to configure a model implementation. The same set of model parameters shall be valid for all views. See 6.11.20.