IEEE1685 & IP-XACT 1.4 Vendor Extensions for Open Core Protocol

Release 1.4

IP-XACT 1.4 & IEEE1685 Vendor Extensions for Open Core Protocol
Document Revision 1.4

In September 2013, Accellera Systems Initiative (Accellera) acquired certain assets of OCP-IP. These assets include the current OCP 3.0 standard and the supporting infrastructure. OCP 3.0 and OCP supplemental materials was released by Accellera in October 2013.

EXCEPT AS OTHERWISE EXPRESSLY PROVIDED, THE META DATA SPECIFICATION IS PROVIDED BY ACCELLERA TO MEMBERS “AS IS” WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS. ACCELLERA SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES OF ANY KIND OR NATURE WHATSOEVER (INCLUDING, WITHOUT LIMITATION, ANY DAMAGES ARISING FROM LOSS OF USE OR LOST BUSINESS, REVENUE, PROFITS, DATA OR GOODWILL) ARISING IN CONNECTION WITH ANY INFRINGEMENT CLAIMS BY THIRD PARTIES OR THE SPECIFICATION, WHETHER IN AN ACTION IN CONTRACT, TORT, STRICT LIABILITY, NEGLIGENCE, OR ANY OTHER THEORY, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Table of contents

1. Overview

1.1 Introduction

This document describes the IEEE 1685 & IP-XACT 1.4 vendor extensions defined by the OCP-IP Metadata Working Group (MDWG). These vendor extensions are used in the OCP 2.2 IP-XACT busDefinition and abstractionDefinition provided by the OCP-IP consortium and they enable to extend the IEEE1685 & IP-XACT 1.4 schema to support the configurability features of the OCP protocol.

The following members of the OCP Metadata Working Group have participated in the definition of the IP-XACT extensions:

Christophe Amerijckx (ST)
Cyril Spasevski (Magillem Design Services)
Kamil Synek (Sonics)
Mark Noll (Synopsys)
Pascal Chauvet (Sonics)
Prashant Karandikar (Texas Instruments),
Stéphane Guntz (Magillem Design Services)
Vesa Lahtinen (Nokia),
Yasuhiko Kurosawa (Toshiba)

OCP vendor extensions schema, examples and associated checkers can be downloaded by OCP-IP members from www.ocpip.org website

1.2 Scope

Current OCP Extensions are defined for IP-XACT schema version 1.4 & IEEE1685/IP-XACT-1.5, inside the OCP busDefinition and abstractionDefinition IP-XACT files. They enable to capture:

- the list of OCP logical parameters and associated configuration compliance rules (in the busDefinition)
- the list of OCP logical ports and configurable constraints on their width and presence (in the abstractionDefinition)

In an OCP IP-XACT component, an OCP interface is defined exactly as a standard IP-XACT interface (i.e. without any additional vendor extension): it contains a port mapping between physical ports and OCP logical ports and a list of OCP parameters for its configuration.

Associated checkers enable to verify the OCP interface configuration compliance. Cross-interface configuration checks are not yet supported by the OCP extensions and checkers.

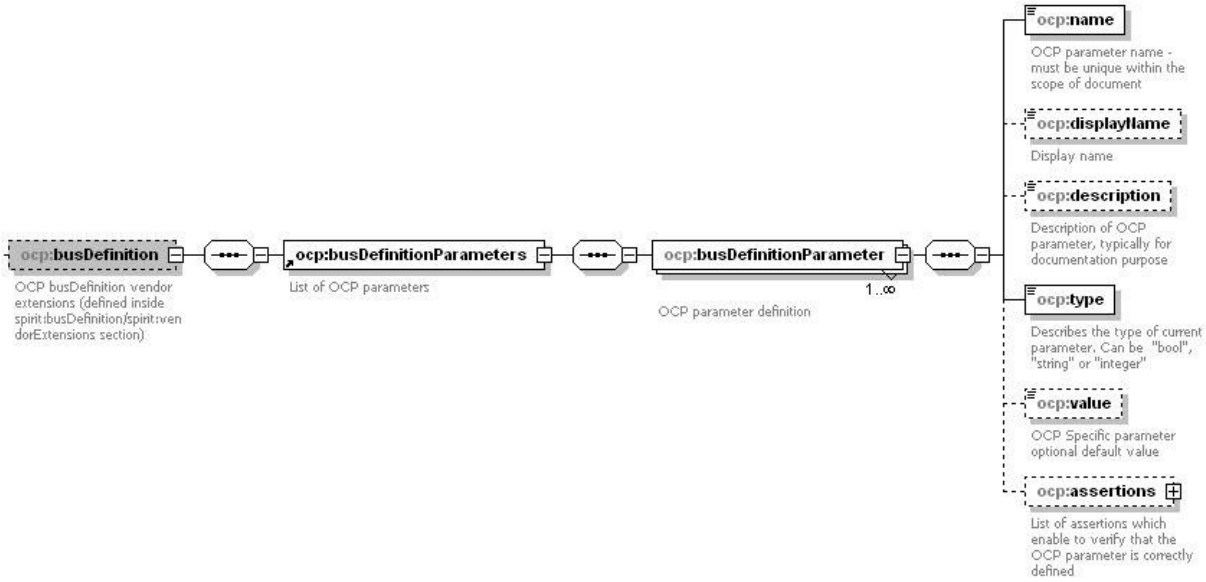
2. busDefinitions Extensions

2.1 busDefinition parameters

The OCP busDefinition contains the list of possible OCP parameters that can be used to configure an OCP interface, with their name, type and optional default value. In specific cases, configuration rules are associated to the parameter definitions, when a parameter value is depending on other parameter values.

The OCP busDefinition parameters are defined in the spirit:vendorExtensions section at the root level of the IP-XACT OCP busDefinition.

2.1.1 Schema



2.1.2 Description

The OCP busDefinition parameters are defined as a sequence of parameters inside a root *ocp:busDefinition/ocp:busDefinitionParameters* element. Elements defined in an *ocp:busDefinitionParameter* are as follows:

a) ***name*** (mandatory) specifies the name of the parameter. The **name** element is of type ***Name***
b) ***displayName*** (optional) allows a short descriptive text to be associated with the containing element. The **displayName** element is of type ***string***.
c) ***description*** (optional) allows a textual description of the containing element. The **description** element is of type ***string***.
d) ***type*** (mandatory) indicates the type of the parameter. The type can be “bool”, “string” or “integer”
e) ***value*** (optional) indicates the default parameter value

f) ***assertions*** (optional) contains the list of configuration rules associated to this parameter. See paragraph 2.2

2.1.3 Example

The following example shows the definition of an OCP parameter named *broadcast_enable* (*boolean* type, default value= *false)* inside the root *spirit:vendorExtensions/ocp:busDefinition/ocp:busDefinitionParameters* section.

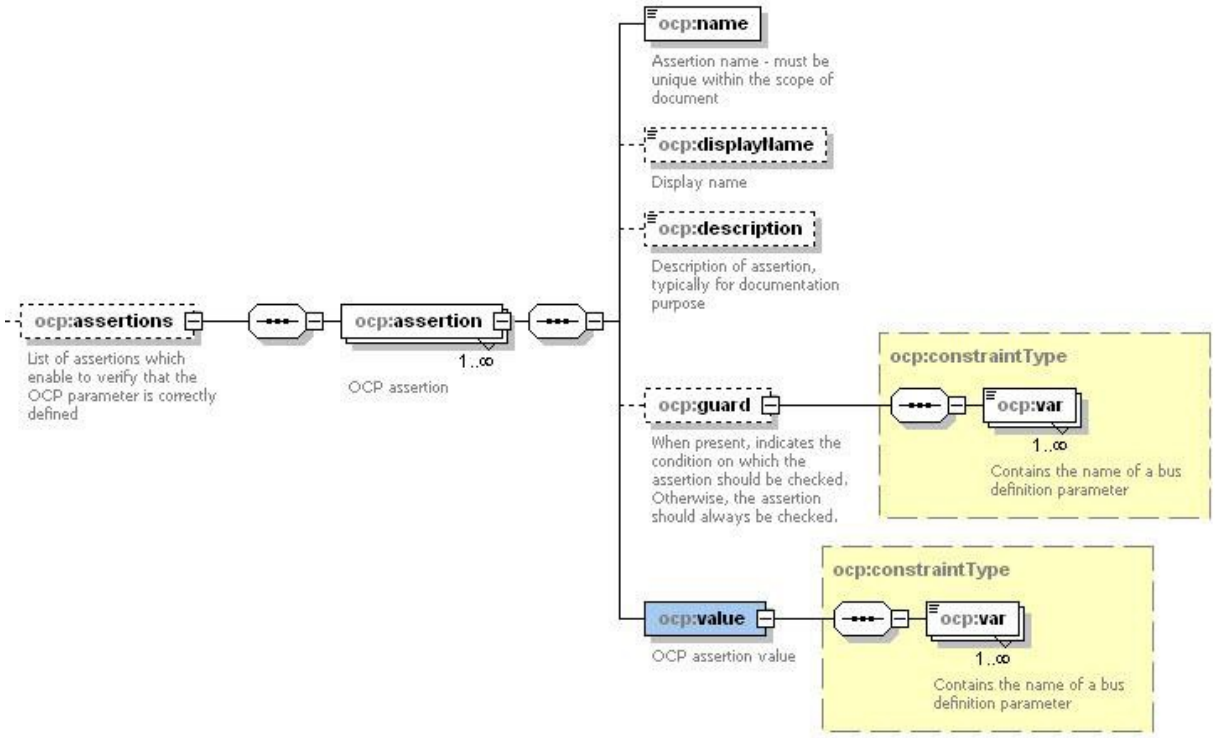
This parameter is also associated to specific configuration rules (defined inside the *ocp:assertions* section), which are not shown here for clarity reasons.

```
<spirit:busDefinition xsi:schemaLocation="http://www.spiritconsortium.org/XMLSchema/SPIRIT/1.4
http://www.spiritconsortium.org/XMLSchema/SPIRIT/1.4/busDefinition.xsd"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xmlns:ocp=http://www.ocpip.org
xmlns:spirit=http://www.spiritconsortium.org/XMLSchema/SPIRIT/1.4 >
        <spirit:vendor>ocpip.org</spirit:vendor>
        <spirit:library>OCP</spirit:library>
        <spirit:name>OCP2_2</spirit:name>
        <spirit:version>1.0</spirit:version>
        <spirit:directConnection>true</spirit:directConnection>
        <spirit:isAddressable>true</spirit:isAddressable>
        <spirit:vendorExtensions>
                <ocp:busDefinition>
                        <ocp:busDefinitionParameters>
                                <ocp:busDefinitionParameter>
                                        <ocp:name>broadcast_enable</ocp:name>
                                        <ocp:description>Enable the command "Broadcast". If the slave has
broadcast_enable set to false, the master must have broadcast_enable set to false.</ocp:description>
                                        <ocp:type>boolean</ocp:type>
                                        <ocp:value>false</ocp:value>
                                        <ocp:assertions>
                                                ...
                                        </ocp:assertions>
                                </ocp:busDefinitionParameter>
                        …
                        </ocp:busDefinitionParameters>
                </ocp:busDefinition>
        </spirit:vendorExtensions>
</spirit:busDefinition>
```

2.2 Assertions

2.2.1 Schema

The *ocp:assertions* section defines the list of configuration rules associated to an OCP busDefinition parameter. Each assertion is defined with a condition on which the assertion should be checked (defined inside *ocp:guard* element), and an assertion formula (defined inside *ocp:value* element).
A parameter defined on an OCP busInterface is considered as correctly configured if all associated assertion computed values resolve to *true*.



2.2.2 Description

Elements defined in an *ocp:busDefinitionParameter/ocp:assertions/ocp;assertion* are as follows:

a) ***name*** (mandatory) specifies the name of the parameter. The **name** element is of type ***Name***

b) ***displayName*** (optional) allows a short descriptive text to be associated with the containing element. The **displayName** element is of type ***string***.

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

d) ***guard*** (optional) indicates the condition on which the assertion should be checked. This condition is defined as a mathematical formula depending on OCP parameters. Parameter names are indicated inside *ocp:var* elements. If the ***guard*** element is not defined, the assertion should always be checked.

e) ***value*** (mandatory) indicates the formula to compute the assertion value. The assertion value is defined as a mathematical formula depending on OCP parameters. Parameter names are indicated inside *ocp:var* elements. If the ***value*** resolves to true, the assertion is considered as verified.

2.2.3 Example

Following assertion defined on *mthreadbusy_exact* parameter defines that this parameter can only be set to *true* if *mthreadbusy* parameter is also set to *true*.

```
<ocp:busDefinitionParameter>
        <ocp:name>mthreadbusy_exact</ocp:name>
        <ocp:description>Define strict protocol semantics for the corresponding phase. mthreadbusy_exact can be set to true only when mthreadbusy is true</ocp:description>
        <ocp:type>boolean</ocp:type>
        <ocp:value>false</ocp:value>
        <ocp:assertions>
                <ocp:assertion>
                        <ocp:name>response_cfg_mthreadbusy_exact_enable_mthreadbusy</ocp:name>
                                <ocp:description>mthreadbusy_exact can only be enabled if mthreadbusy is enabled.</ocp:description>
                        <ocp:value><ocp:var>mthreadbusy</ocp:var>='true'</ocp:value>
                </ocp:assertion>
        </ocp:assertions>
</ocp:busDefinitionParameter>
```

The following examples defines that:

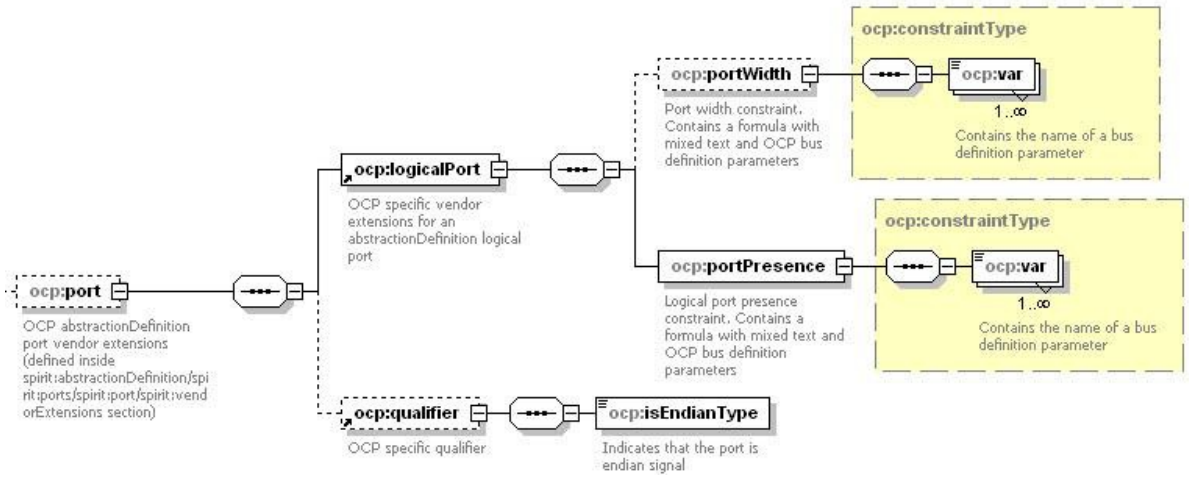
- *addr_wdth* assertion should only be checked if *addr* parameter is set to *true*
- the assertion formula is: *addr_wdth greater than or equal to max (1, ceiling(log(2,data_wdth-3)))*

```
<ocp:busDefinitionParameter>
        <ocp:name>addr_wdth</ocp:name>
        <ocp:description>Size of MAddr</ocp:description>
        <ocp:type>integer</ocp:type>
        <ocp:assertions>
            <ocp:assertion>
                <ocp:name>request_cfg_addr_width_depends_data_wdth</ocp:name>
                <ocp:description>addr_wdth defines a minimum addr_wdth value that is based on the data bus byte width, and is defined as: min_addr_wdth = max(1, ceil(log2(data_wdth))-3)</ocp:description>
                <ocp:guard><ocp:var>addr</ocp:var>='true'</ocp:guard>
<ocp:value><ocp:var>addr_wdth</ocp:var> &gt;= max((1,ceiling(spirit:log(2,<ocp:var>data_wdth</ocp:var>)) - 3))
</ocp:value>
            </ocp:assertion>
        </ocp:assertions>
</ocp:busDefinitionParameter>
```

3. AbstractionDefinition extensions

The OCP abstractionDefinition extensions are defined in the *spirit:vendorExtensions* section of an IP-XACT logical port. These extensions enable to define a configurable presence of this logical port on an OCP interface (inside the *ocp:port/ocp:logicalPort/ocp:portPresence* element) or a configurable width (inside the *ocp:port/ocp:logicalPort/ocp:portWidth* element), which are depending on OCP parameter values.

3.1 Schema



3.2 Description

The OCP abstractionDefinition port extensions, contained in a root ***ocp:port*** element, are defined as follows:

a) ***logicalPort*** (mandatory) which contains 2 sub-elements: ***portWidth*** (optional) and ***portPresence (***mandatory). These 2 elements are used to defined a configurable presence and width of the logical port, using formulas similary as for the parameter assertions
b) ***qualifier*** (optional) which contains a unique ***isEndianType*** element (***boolean*** type), indicates that the physical port mapped to this logical port is defining the interface endianness value.

3.3 Example

Following example shows the definition of *MAddr* logical port, which is also associated to OCP extensions. These extensions define that:

- The physical port mapped to *MAddr* logical port should have a width equal to the *maddr_wdth* parameter value
- *MAddr* logical port should only be used on an OCP interface if *addr* parameter value is set to *true*

```
<spirit:port>
        <spirit:logicalName>MAddr</spirit:logicalName>
        <spirit:description>The Transfer address, MAddr specifies the slave-dependent address of the resource
targeted by the current transfer. To configure this field into the OCP, use the addr parameter. To configure the width
of this field, use the addr_wdth parameter. MAddr is a byte address that must be aligned to the OCP word size
(data_wdth). data_wdth defines a minimum addr_wdth value that is based on the data bus byte width, and is defined
as: min_addr_wdth = max(1, ceil(log2(data_wdth))-3)
If the OCP word size is larger than a single byte, the aggregate is addressed at the OCP word-aligned address and
the lowest order address bits are hardwired to 0. If the OCP word size is not a power-of-2, the address is the same as
it would be for an OCP interface with a word size equal to the next larger power-of-2.
        </spirit:description>
                <spirit:wire>
                        <spirit:qualifier>
                                <spirit:isAddress>true</spirit:isAddress>
                        </spirit:qualifier>
                        <spirit:onMaster>
                                <spirit:presence>optional</spirit:presence>
                                <spirit:direction>out</spirit:direction>
                        </spirit:onMaster>
                        <spirit:onSlave>
                                <spirit:presence>optional</spirit:presence>
                                <spirit:direction>in</spirit:direction>
                        </spirit:onSlave>
                        <spirit:defaultValue>0</spirit:defaultValue>
                </spirit:wire>
                <spirit:vendorExtensions>
                        <ocp:port>
                                <ocp:logicalPort>
                                        <ocp:portWidth>
                                                <ocp:var>addr_wdth</ocp:var>
                                        </ocp:portWidth>
                                        <ocp:portPresence>
                                                <ocp:var>addr</ocp:var>='true'
                                        </ocp:portPresence>
                                </ocp:logicalPort>
                        </ocp:port>
                </spirit:vendorExtensions>
</spirit:port>
```

4. OCP component example

ocpip.org_OCP_OCP_component_correct_configuration_1.0.xml example contains the definition of an OCP component, defined with one OCP interface. This interface is defined with a standard IP-XACT mapping between the physical ports of the component and the logical ports defined on the OCP abstractionDefinition.

In addition, the interface is associated with a list of OCP parameters, whose values match with the OCP signals present on the OCP interface. If you apply the verification checkers on the interface, they confirm that the interface configuration is correct, according to the rules defined in the OCP abstractionDefinition and busDefinition.

See ocpip.org_OCP_OCP_component_correct_configuration_1.0.xml file for more details on how the interface is defined and configured.

5. Interface Configuration Checker

OCP Interface Configuration Checker is based on Vendor Extensions defined in sections above. It uses xslt transforms in 1.0 style sheet to support most of the tools.

The Interface configuration checker checks each OCP interface checker with following VLNV (ocpip.org , OCP , OCP2_2 , 1.0) for bus Type and (ocpip.org , OCP ,OCP2_2_RTL , 1.0) for abstraction Type.

The usage of the checker would vary depending on XSLT processor used. Sample usage of the checker with AltovaXML is shown below.

```
AltovaXML /xslt1 OCPChecker_IEEE1685.xsl /in
ocpip.org_OCP_OCP_component_incorrect_configuration_1.0.xml

Interface OCP_master_busIF_correct_configuration
   Port width constraint is not verified for port MData
   Port width constraint is not verified for port SData
   Port presence constraint is not verified for port MReset_n
```

In above example data_wdth parameter is set to 63 and hence port width constraint for MData & SData port has failed since those 64 bit physical data ports.

mreset parameter is set to false hence MReset_n presence as component interface is not justified.

For correct configuration no messages would appear.

6. References

1. IP-XACT v1.4: A specification for XML meta-data and tool interfaces
2. IP-XACTv1.4 schema http://www.spiritconsortium.org/XMLSchema/SPIRIT/1.4
3. IEEE1685-2009 : IEEE Standard for IP-XACT , Standard Structure for Packaging , Integrating , and reusing IP within Tool Flows
4. IEEE 1685-2009 schema http://www.spiritconsortium.org/XMLSchema/SPIRIT/1685-2009
5. Open Core Protocol Specification Release 2.2

OCP-IP Administration
3116 Page Street
Redwood City, CA 94063
Ph: +1 (512) 551.3377
Fax: +1 (650) 365.4658
admin@ocpip.org
www.ocpip.org