Documentation of Issues
For
IEEE 1685™: IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows

This document contains a running list of issues and considerations to be addressed in the next version of the standard. The issues and considerations listed here are collected and managed by the Accellera IP-XACT Working Group. IEEE 1685 can be downloaded here.

Issues

- Incorrect wording for SCR about clocks being on scalar ports only – 12.9 – should say scalar or single-bit

- Annex I uses 0x<digits> for all hex numbers which is ILLEGAL

- Description of whiteBoxType within whiteBoxElement indicates it can be set to 'register' which is no longer supported. This is in the schema, not the standard document.

- Problems with TGI.xlsx (sent to Erwin on 10/31/14). These also appear this way in the standard and need to be updated accordingly.

    1) The callback getRegisterFileTypeIdentifier is incorrectly defined twice. The second definition should have been named set* instead of get*

    2) The callback getAbstractorViewComponentInstantiationRef is defined twice. Once returning componentInstantiationName and once returning componentInstantiationID. I assume the latter is the desired definition since you can get the name from the ID.

    3) The callback removeFileSetDependency is defined twice, once for files and once for fileSets. Seems like it should have a single definition with the input argument being "fileSetID | fileID" to be consistent with other callbacks that supported multiple ID types.

    4) The callbacks getFileBuilderFileType, getFileBuilderCommand, getFileBuilderFlags, getFileBuilderReplaceDefaultFlags, setFileBuilderFlags, setFileBuilderReplaceDefaultFlags are all defined twice. Once in the address space operations section and then an identical definition in the fileBuilder section. Since these are fileBuilder operations it seems like the definitions within the address space section are redundant.

- SCR addressBlockContent is a definition, not an SCR. This needs to either be removed or potentially updated to indicate that when the type is 'reserved' we should not allow registers.

- Link in SCR 7.19 is not to the correct location.

- Multiple references to the now obsolete 'opaque' attribute. Also need to update documentation about opaque bridges to indicate how they are now modeled with sub-space maps.

- 6.11.3.2.e (register field within register) is incorrectly documented as being optional. At least one is required in the current schema.

- Leon expressions (as in uart.xml) are in some cases invalid when combining the '$pow' function with the '%' operator as the former returns a real and the latter requires integer arguments. Up-conversion is currently broken in this regard.

- In Table F.1, in the "Example" column for the "Set" row, the text should be "Set parameter value" instead of "Get parameter Value".

- There is no clear definition about the difference between the Base and Extended API categories.

- The tables 7.2.1 and 7.2.2 don't have any descriptive text. This should either be fixed or more likely they should merge into the field operations table(s).

- Both the IEEE 1685-2009 and 1685-2014 Std documents incorrectly state that an enumeration value for TestContraint is "unConstrained" (with upper case C), page 116 and 93 respectively. The XML schemas (memoryMap.xsd) correctly state that the value is "unconstrained" (with lower case C). Use of the incorrect value will result in syntax errors during schema validation and are likely not to work in compliant tools. The next revision of the standard will be changed to use the correct all-lower-case name.

- **replaceDefaultFlags**: Section 6.15.4 describes replaceDefaultFlags backwards relative to the schema.

- **Examples in IEEE1685-2014 LRM reference wrong schema**
  The examples in Annex I use the namespace *http://www.accellera...hema/IPXACT/2.0* instead of *http://www.accellera...PXACT/1685-2014*

- **XML for Annex I.5 isn't well formed**
  The *designConfigurations* tag isn't properly closed. It should be:
  <ipxact:designConfigurations>
  <ipxact:ipxactFile>
  <ipxact:vlnv vendor="accellera.org" library="Sample"
  name="SampleDesignConfiguration" version="1.0"/>
  <ipxact:name>./SampleDesignConfiguration.xml</ipxact:name>
  </ipxact:ipxactFile>

</ipxact:designConfigurations>

- **Typo in Annex I.6 remap state name.**
  The second remap state is called 'Nornmal', but should be 'Normal'.

- **Bibliography references non-existing URL**
  B12 lists http://www.accellera.../refs/toolnames as the source for tool names compatible with the *envIdentifier* field. The URL doesn't exist.

- **Example in Annex I.6 lists same file twice in fileSet**
  <ipxact:name>VerilogFiles</ipxact:name>
  <!-- LINK: file: see 6.15.2, file -->
  <ipxact:file>
  <ipxact:name>../src/component.v</ipxact:name>
  <ipxact:fileType>verilogSource</ipxact:fileType>
  <ipxact:isStructural>true</ipxact:isStructural>
  </ipxact:file>
  <ipxact:file>
  <ipxact:name>../src/component.v</ipxact:name>
  <ipxact:fileType>verilogSource</ipxact:fileType>
  </ipxact:file>
  </ipxact:fileSet>

- **Obsolete TODO reference in annex I.6**
  There is a TODO comment in the example component:
  <!-- TODO: MISSING definition of resetType in document -->

- **XML for Annex I.7 isn't well formed**
  There is a comment in the sample design: <!-- Export Master interface -- will be used for TLM to RTL conversion -->
  According to the XML specification, it's illegal to have '--' inside comments (except when followed immediately by '>'), which causes parsing to fail.

- **XML for Annex I.9 is not well formed**
  The </ipxact:generatorChain> closing tag is missing at the end of the file.

- **Section C.6.2 describing configurableLibraryRefType is incomplete**
  The description in part (e) for configurableElementValues needs to be expanded to cover the other element types where it is now valid. The new wording should be:

  configurableElementValues (optional) specifies the configuration for a specific component instance, bus type, abstraction type, design instantiation, design configuration instantiation, or generator chain configuration by providing the value of a specific parameter. See C.5.

- **Section 6.10.2.2 describing remapPort element type is not correct**

  remapPorts (optional) contains a list of **remapPort** elements. **remapPort** (mandatory) specifies when the remap state gets effective. A collection of **remapPort** elements make up the condition for this remap state. The **remapPort** element contains the logical value of the single port bit specified by the following attributes:

  1) **portRef** (mandatory; type: ***portName***) attribute is the name of the port in the containing description to which this logic value comparison is assigned. See 6.12.7.
  2) **portIndex** (optional; type: ***unsignedIntExpression*** (see C.3.7)) attribute references the index of a port in the containing description, when the port being referenced is vectored.
  3) **value** (mandatory; type: ***unsignedIntExpression*** (see C.3.7)) is the value necessary so the specified port activates the **remapState**.

  All **remapPort value** elements shall be true for the remap state to be enabled.

- **Section 6.11.9.2 describing modifiedWriteValue is not complete for oneToClear, oneToSet, oneToToggle, zeroToClear, zeroToSet, and zeroToToggle. Description of opposite bit value is missing.**

  modifiedWriteValue (optional) element to describe the manipulation of data written to a field. The value shall be one of oneToClear, oneToSet, oneToToggle, zeroToClear, zeroToSet, zeroToToggle, clear, set, or modify. If the **modifiedWriteValue** element is not specified, the value written to the field is the value stored in the field.
  oneToClear means in a bitwise fashion each write data bit of a one shall clear (set to zero) the corresponding bit in the field, and each write data bit of a zero shall not effect that bit.
  oneToSet means in a bitwise fashion each write data bit of a one shall set (set to one) the corresponding bit in the field, and each write data bit of a zero shall not effect that bit.
  oneToToggle means in a bitwise fashion each write data bit of a one shall toggle the corresponding bit in the field, and each write data bit of a zero shall not effect that bit.
  zeroToClear means in a bitwise fashion each write data bit of a zero shall clear (set to zero) the corresponding bit in the field, and each write data bit of a one shall not effect that bit.
  zeroToSet means in a bitwise fashion each write data bit of a zero shall set (set to one) the corresponding bit in the field, and each write data bit of a one shall not effect that bit.
  zeroToToggle means in a bitwise fashion each write data bit of a zero shall toggle the corresponding bit in the field, and each write data bit of a one shall not effect that bit.
  clear means after a write operation all bits in the field are cleared (set to zero).
  set means that after a write operation all bits in the field are set (set to one).
  modify means that after a write operation all bits in the field may be modified in an undefined way. In this situation, the modify attribute can be set to a user-defined value to provide additional detail.

- **Section C.21.2 describing pathSegment indices is incorrectly mentioning that the indices apply to IP-XACT objects. The indices apply to language-specific objects.**

  Description
  The **pathSegments** element specifies an ordered list of **pathSegment** elements. A **pathSegment** is one node in the hierarchical path. When concatenated with a desired separator, the elements in this form a language-specific path for the parent slice into the referenced view. The **pathSegment** element contains the following elements:
  a) **pathSegmentName** (mandatory; type: ***string***) is one node in the path.
  b) **indices** (optional) specifies a list of **index** elements. The **indices** specify an element in a language-specific object to which the encapsulating **accessHandle** applies. See C.9.

Considerations

- Limited range of tied values (64 bits) constrains the width of busses that can be tied.

- **New SCR [driverPortMapCondition]**
  If a logical port has requiresDriver set to true and driverType set to clock, then a component port with a driver mapped to such a logical port shall have a clockDriver. If a logical port has requiresDriver set to true and driverType set to singleShot, then a component port with a driver mapped to such a logical port shall have a singleShotDriver.

- Add meaning of address block width. Current interpretation is data path width. Add SCR to forbid misalignment of registers in address blocks (i.e., a 32-bit register with offset 3 in a 32 bit wide address block in an 8 bit address unit memory map). How to handle registers that are wider than address block (64 bit registers in 32 bit address block).

- Remove SCR 10.5 **[HierFamilyBusIntfPortMapCondition]**
  If any member of a hierarchical family of bus interfaces has a **portMap** subelement, they all shall.

- **F.7.25.5 addAbstractionTypePortMap has ambiguous input argument**
  Description: Add a portMap with the given name, logicalPortName, and physicalPortNameOrTieValue to the given abstractionType
  Returns: portMapID of type *String* - Handle to a new portMap element
  Input: abstractionTypeID of type *String* - Handle to an abstractionType element
  Input: logicalPortName of type *String* - Logical port name
  Input: physicalPortNameOrTieValue of type *String* - Physical port name or logical tie off value

  This TGI call should be splitted in:
  F.7.25.5a addAbstractionTypePortMap
  Description: Add a portMap with the given name, logicalPortName, and physicalPortName to the given abstractionType
  Returns: portMapID of type *String* - Handle to a new portMap element
  Input: abstractionTypeID of type *String* - Handle to an abstractionType element
  Input: logicalPortName of type *String* - Logical port name
  Input: physicalPortName of type *String* - Physical port name

  F.7.25.5b addAbstractionTypePortMapWithTieOff
  Description: Add a portMap with the given name, logicalPortName, and logicalTieValue to the given abstractionType
  Returns: portMapID of type *String* - Handle to a new portMap element
  Input: abstractionTypeID of type *String* - Handle to an abstractionType element
  Input: logicalPortName of type *String* - Logical port name
  Input: logicalTieValue of type *String* - Logical tie off value

- **Section 6.11.2.2 describing register dim and Section 6.11.6.2 describing registerFile dim are not correct. Value 0 should not be legal.**

  dim (optional type: unsignedPositiveLongintExpression (see C.3.10)) assigns an unbounded dimension to the register, so it is repeated as many times as the value of the dim elements. For multi-dimensional register arrays, the memory layout is presumed to follow the IEEE Std 1666™ [B4] (SystemC) language rules.