IP-XACT Training

Agenda

- Challenges in Design/Verification
- Introduction to IP-XACT
- IP-XACT Constructs
- IP-XACT Design Environment
- Component
  - Component Element
  - addressSpace
  - addressBlock
  - addressBlockDefinitionGroup
  - Register
  - registerDefinitionGroup
  - Alternate Registers
  - registerFile
  - registerFileDefinitionGroup
  - Field
  - Memory Maps
  - Memory Remap
  - Memory Banks
  - Banked address block
  - Banked bank
- Model
  - Model Structure
- Design
  - Design Element Information
  - Component Instances
  - ad hoc connections
- Interface Definition
  - Protocol Descriptions
  - Bus Definition
  - Abstraction Definition
  - Bus Interface
- Generator Chain
- Design Configuration
- Vendor Extensions
- Support in IDesignSpec
  - Properties in IDS
  - Properties & IDS Properties
- Advanced Topics
  - File Sets
  - Enumerated Values
  - Comparison between SystemRDL & IP-XACT
- Tight Interface Generator
  - TGI APIs
  - TGI Use Case
- IP-XACT Use Case Scenario
  - IP-XACT as an output (IDS-Integrate)
  - IP-XACT as an output (IDesignSpec)
  - IP-XACT as an input (IDS-Integrate)
- Benefits of using IP-XACT

Typical Challenges in Design/Verification

- **Ever increasing design complexity**
  - IP integration
  - Verification
- **Testbench Development**
  - Components Integration
  - IP configurations
- **Register Testcase Development**
- **Unavoidable changes during design process**
  - Register definition, location, type, or implementation
  - Monotonous work
- **Maintaining consistency within the teams**

What are the solutions ?

- Single Specification for all information
- Single description for all registers
- All representations generated from single source
- Automated workflow

Introduction to IP-XACT < />

- The main purpose of IP-XACT is basically for delivering IPs from IP providers to IP users
- IP-XACT has metadata that documents the characteristics of an Intellectual Property(IP) required for the automation and to define an Application Programming Interface (API) to make this directly accessible to automation tools or the users
- It provides a way to IP providers to package the information like, ports and interfaces, hw/sw memorymaps, and the files which constitutes the IP(fileset)
- This standard was created by the Spirit Consortium and is now part of Accellera Systems Initiative
- This training focuses on providing an opportunity to learn more about IP-XACT and how this standard can be used to enhance an IP based design and verification flow

IP-XACT Constructs < />

- An IP-XACT description of a design or component consists of a set of XML documents referring to one another

- Main document types are:
  - *Component:* A description of 3 component type, including interfaces, memory maps, and registers (IP)
  - *Bus Definition:* A description of a bus type
  - *Design:* A high level description of a design (SoC Netlist)

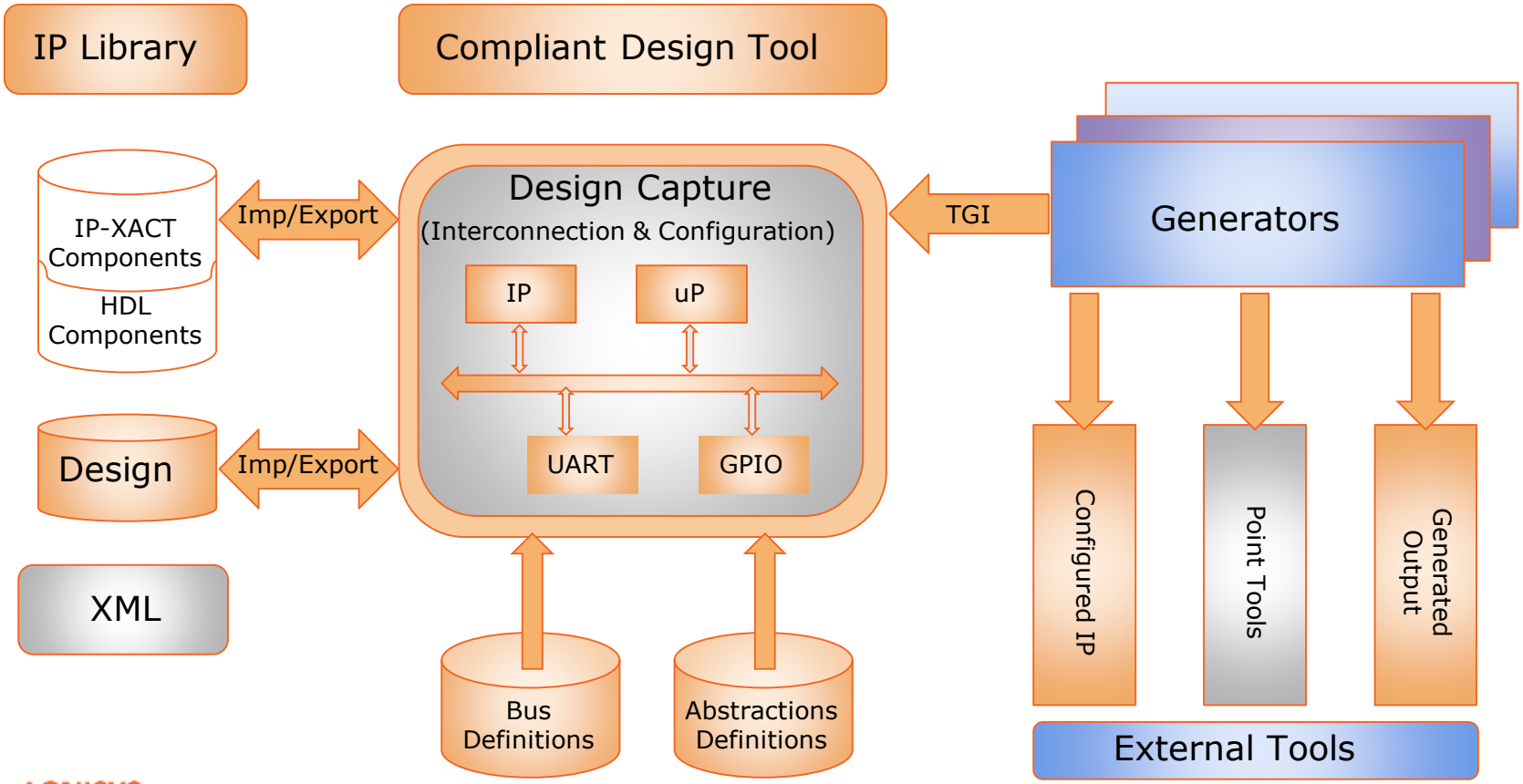
- References between IP-XACT document are by 4 element identifier (vendor, library, name and version, often abbreviated to ***VLNV***)

```
<ipxact:vendor>Agnisys</ipxact:vendor>
<ipxact:library>mixed_signals</ipxact:library>
<ipxact:name>top</ipxact:name>
<ipxact:version>1.0</ipxact:version>
```

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

IP-XACT Design Environment

Design Environment



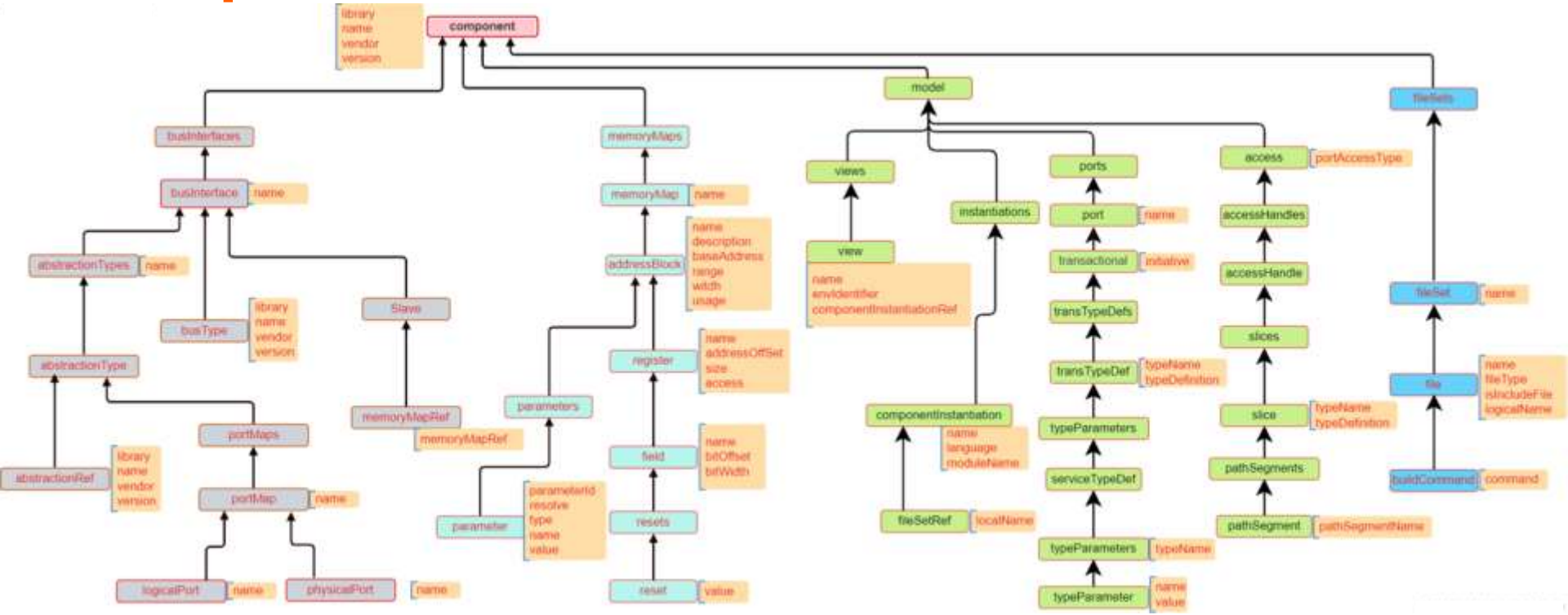
Design Environment 

- A Design Environment enables the designer to work with IP-XACT design through a coordinated front-end and design database
- These tools create and manage the top-level meta-description of system design and may provide two basic types of services:
    - *design capture*: Expression of design configuration by the IP provider and design intent by the IP user
    - *design build*: Creation of a design to those intentions
- As part of design capture, a system design tool recognizes the structure and configuration options of imported IP
    - *Structure*: It implies both the structure of the design as well as the structure of the IP package
    - *Configuration*: It is the set of options for handling the imported IP
- As part of design build, generators may be provided internally by a system design tool to achieve the required IP integration or configuration, or they may be provided by an IP provider and launched by the system design tool

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

IP-XACT Component

Component



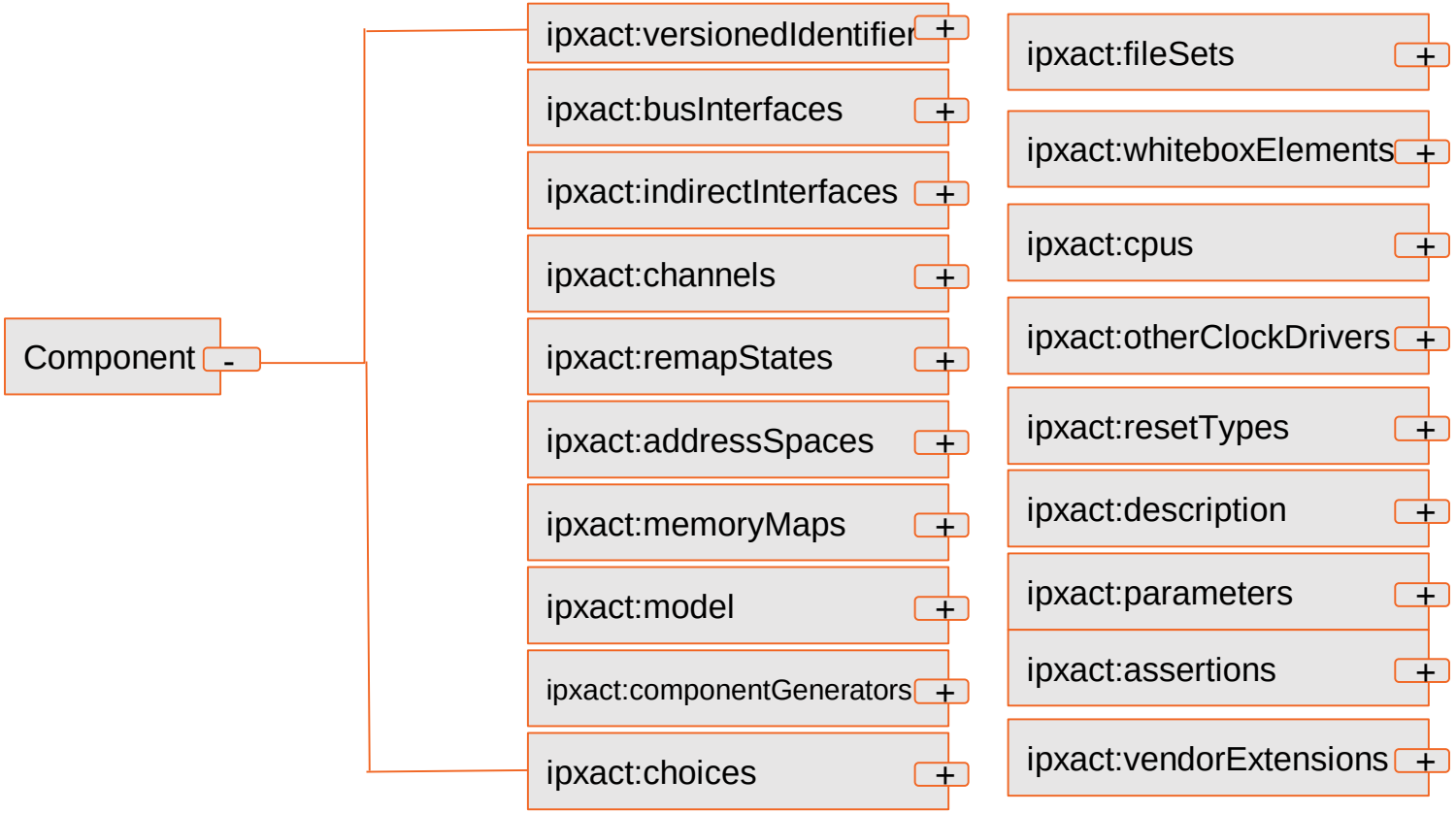
Component

- An IP-XACT component is used to describe the meta-data associated with any IP that can be instantiated in a design
- A Component contain, VLNV desc, Bus Interfaces, AddressSpaces, MemoryMaps, Models, Parameters and, vendorExtensions
- Purpose: To enable the (re-)use of an IP through IP-XACT without the need for information from the implementation files

```
<ipxact:component xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:soce="http://www.agnisys.com/">
    <ipxact:vendor>Agnisys</ipxact:vendor>
    <ipxact:library>mixed_signals</ipxact:library>
    <ipxact:name>top</ipxact:name>
    <ipxact:version>1.0</ipxact:version>
    <ipxact:busInterfaces> [23 lines]
    <ipxact:model> [15 lines]
    <ipxact:parameters/>
</ipxact:component>
```

Discussed Later

Component Element Information



Component Element Information

- The IP-XACT component contains the following **mandatory** and optional elements:
  - **versionedIdentifier**: It provides a unique identifier; it consists of four subelements for a top-level IP-XACT element
  - busInterfaces: It specifies all the interfaces for this component
  - indirectInterfaces: It specifies access points and access information for any indirect memory map
  - channels: It specifies the interconnection between interfaces inside of the component
  - remapStates: It specifies the combination of logic states on the component ports and translates them into a logical name for use by logic that controls the defined address map
  - addressSpaces: It specifies the addressable area as seen from busInterfaces with an interface mode of master or from cpus
  - memoryMaps: It specifies the addressable area as seen from busInterfaces with an interface mode of slave
  - model: It specifies all the different views, ports, and model configuration parameters of the component
  - componentGenerators: It specifies a list of generator programs attached to this component
  - choices: It specifies multiple enumerated lists

Component Element Information

- fileSets: It specifies groups of files and possibly their function for reference by other sections of this component description
- whiteboxElements: It specifies all the different locations in the component that can be accessed for verification purposes
- cpus: It indicates this component contains programmable processors
- otherClockDrivers: It specifies any clock signals that are referenced by implementation constraints, but are not external ports of the component
- resetTypes: It specifies user-defined reset types referenced within field reset elements
- description: It allows a textual description of the component
- parameters: It describes any parameter that can be used to configure or hold information related to this component
- assertions: It contains a list of expressions defining the allowed parameter values
- vendorExtensions: It contains any extra vendor-specific data related to the component.

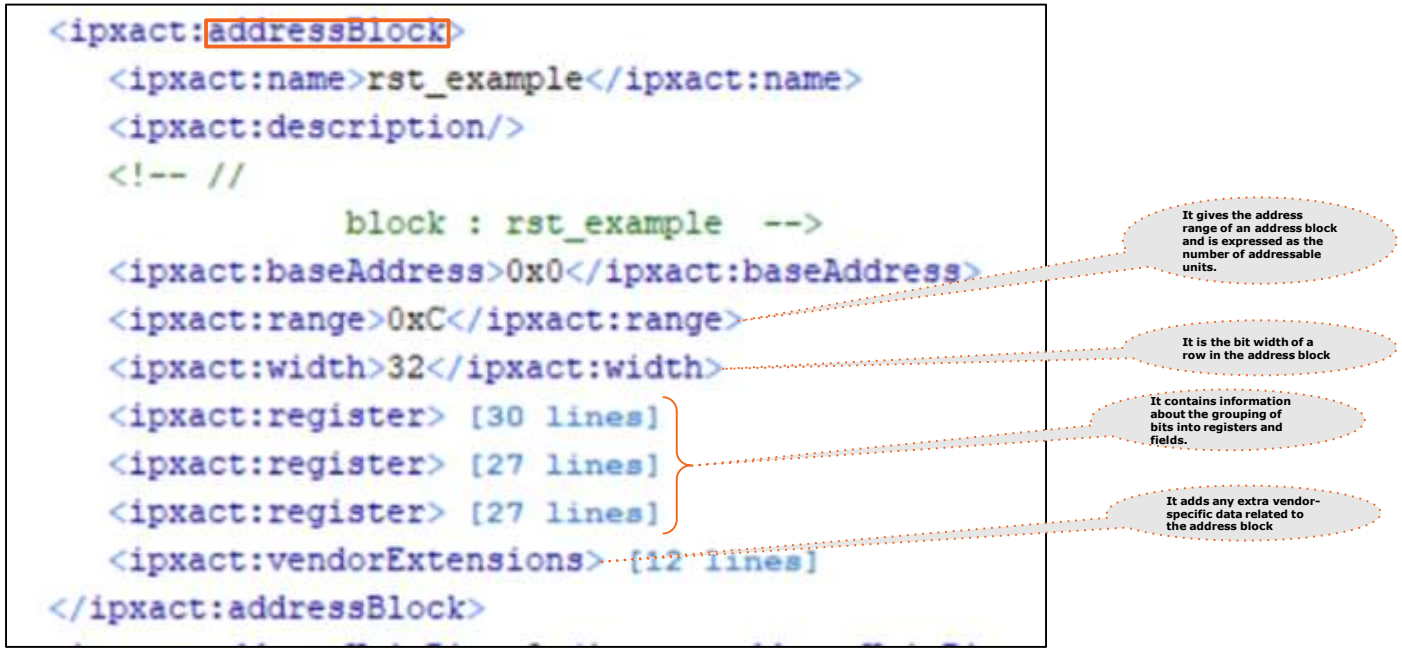
Address Space

- Each addressSpace element defines a logical address space seen by a master bus interface
- An addressSpace contains Name, Range, Width, Segments, executableImage, localmemoryMap

```
<ipxact:addressSpace>
    <ipxact:name>master_0</ipxact:name>
    <ipxact:displayName>region</ipxact:displayName>
    <ipxact:description>Data #0 region</ipxact:description>
    <ipxact:range>'h0800000</ipxact:range>
    <ipxact:width>32</ipxact:width>
</ipxact:addressSpace>
```

Address Block

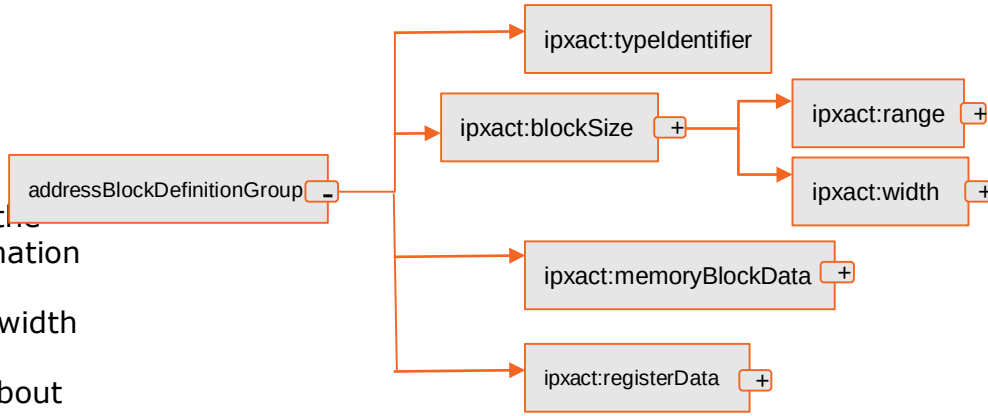
- addressBlock is a contiguous block of memory that is part of memoryMap
- addressBlock contain multiple elements like, Name, Description, blockSize, usage, volatile, access, registerData, registerData and, vendorExtensions



addressBlockDefinitionGroup

- The addressBlockDefinitionGroup group describes the definition information about address blocks

- It contains the following **mandatory** and optional elements:
  - typeIdentifier - It indicates multiple address block elements with the same typeIdentifier in the same description contain the exact same information for the elements
  - **blockSize** - It gives the address range and bit width of a row of an address block.
  - **memoryBlockData** - It contains information about usage, access, volatility, and other parameters
  - **registerData** - It contains information about the grouping of bits into registers and fields



Register

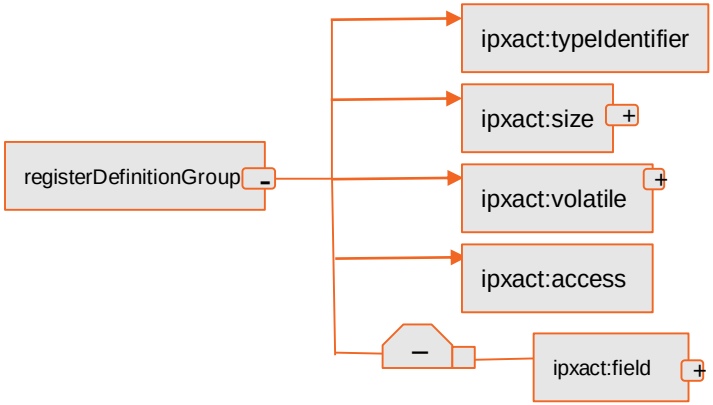
- A register element describes a register in an address block or register file. The bits in the register are numbered from size–1 down to 0
- It contains multiple elements like, Name, Description, dim, addressOffset, size, Volatile, Access, Field, Parameter and, vendorExtensions
- Register can have a *dim element* describing the dimension of the register. If dim is not described, then its value defaults to 1
- The size of a register describes the number of bits in the register. A register size cannot exceed the width of a containing addressBlock

```
<ipxact:register>
  <ipxact:name>Reg1</ipxact:name>
  <ipxact:addressOffset>0x0</ipxact:addressOffset>
  <ipxact:size>32</ipxact:size>
  <ipxact:volatile>true</ipxact:volatile>
  <ipxact:field>
    <ipxact:name>Fld1</ipxact:name>
    <ipxact:bitOffset>0</ipxact:bitOffset>
    <ipxact:resets>
      <ipxact:reset>
        <ipxact:value>0x01</ipxact:value>
        <ipxact:mask>0xFFFFFFFF</ipxact:mask>
      </ipxact:reset>
    </ipxact:resets>
    <ipxact:bitWidth>32</ipxact:bitWidth>
    <ipxact:volatile>true</ipxact:volatile>
    <ipxact:access>read-write</ipxact:access>
    <ipxact:vendorExtensions>
      <ids_properties>
        <resetsignal>sync_rst</resetsignal>
      </ids_properties>
    </ipxact:vendorExtensions>
  </ipxact:field>
  <ipxact:vendorExtensions>
    <ids_properties>
      <hard_reset>false</hard_reset>
      <address>0x0</address>
    </ids_properties>
  </ipxact:vendorExtensions>
</ipxact:register>
```

registerDefinitionGroup

- The registerDefinitionGroup describes the register definition information

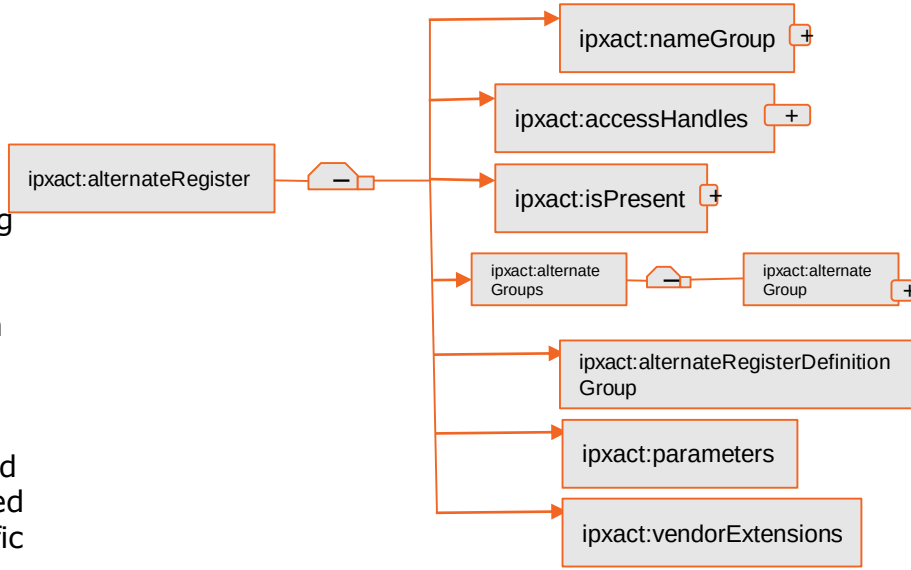
- It contains the following **mandatory** and optional elements:
  - typeIdentifier - It indicates multiple register elements with the same typeIdentifier in the same description contain the exact same information for the elements in the registerDefinitionGroup.
  - size: It is the width of the register, counting in bits.
  - volatile: When true, it indicates in the case of a write followed by read, or in the case of two consecutive reads
  - access: It indicates the accessibility of the register. If this is not present, the access is inherited from the containing addressBlock.
  - field: It describes a bit field within a register



Alternate Registers

- Each alternateRegister element contained within the same alternateRegisters element provides an alternate description for the containing register element

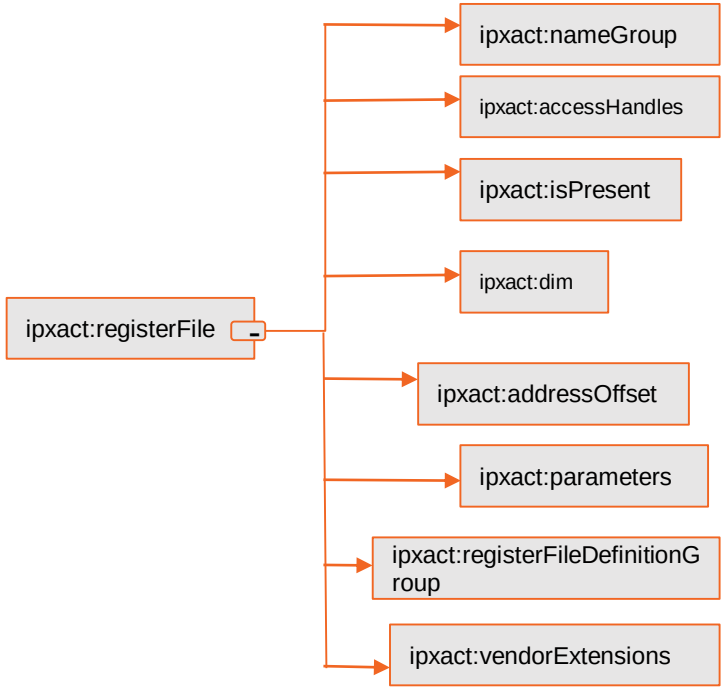
- It contains the following **mandatory** and optional elements:
  - **nameGroup**: The name element shall be unique within the containing alternateRegister element.
  - accessHandles: It specifies view-dependent naming for this register within the corresponding RTL
  - **alternateGroups**: It defines an unbounded list of grouping names to which this alternate description belongs.
  - **alternateRegisterDefinitionGroup**: It describes additional elements for an alternate register
  - parameters: It describes any parameter names and types when the register width can be parameterized
  - vendorExtensions: It adds any extra vendor-specific data related to this register



Register File

- A registerFile element describes a grouping of registers in an address block or register file

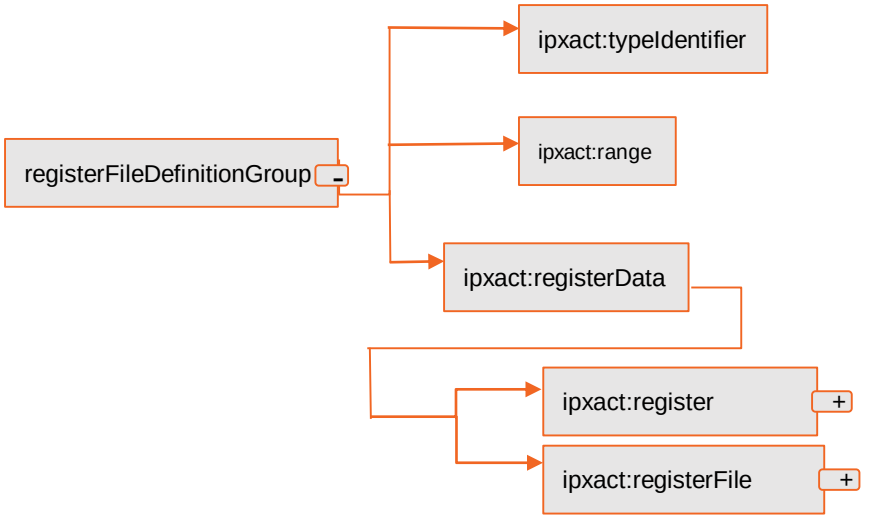
- It contains the following **mandatory** and optional elements:
  - **nameGroup** - The name element shall be unique within the containing registerFile element.
  - accessHandles - It specifies view-dependent naming for this register files within the corresponding RTL
  - dim - Assigns an unbounded dimension to the register
  - **addressOffset** - It describes the offset from the start of the containing addressBlock or registerFile element
  - **registerFileDefinitionGroup** - It describes additional elements for a register file
  - parameters - Describes any parameter names and types when the register width can be parameterized
  - vendorExtensions - Adds any extra vendor-specific data related to this register.




```
<ipxact:registerFile>
   <ipxact:name>CHANNEL</ipxact:name>
   <ipxact:description>Channel Descriptor Registers</ipxact:description>
   <ipxact:dim>4</ipxact:dim>
   <ipxact:addressOffset>'h200</ipxact:addressOffset>
   <ipxact:range>'h10</ipxact:range>
   ...
</ipxact:registerFile>
```

registerFileDefinitionGroup

- The registerFileDefinitionGroup may appear as an element inside the register element
- This element describes the reset value of the register.
- It contains the following **mandatory** and optional elements:
  - typeIdentifier: It indicates multiple register elements with the same typeIdentifier in the same description contain the exact same information for the elements in the registerDefinitionGroup
  - **range**: It gives the range of a register file
  - **registerData**: It contains information about the grouping of bits into registers and fields



Field

- Field element of a register describes a smaller bit field of a register
- A field contains multiple elements like, Name, Description, bitOffset, Resets and, bitwidth

```
<ipxact:field>
   <ipxact:name>Fld1</ipxact:name>
   <ipxact:bitOffset>0</ipxact:bitOffset>
   <ipxact:resets>
      <ipxact:reset>
         <ipxact:value>0x01</ipxact:value>
         <ipxact:mask>0xFFFFFFFF</ipxact:mask>
      </ipxact:reset>
   </ipxact:resets>
   <ipxact:bitWidth>32</ipxact:bitWidth>
   <ipxact:volatile>true</ipxact:volatile>
   <ipxact:access>read-write</ipxact:access>
   <ipxact:vendorExtensions>
      <ids_properties>
         <resetsignal>sync_rst</resetsignal>
      </ids_properties>
   </ipxact:vendorExtensions>
</ipxact:field>
```

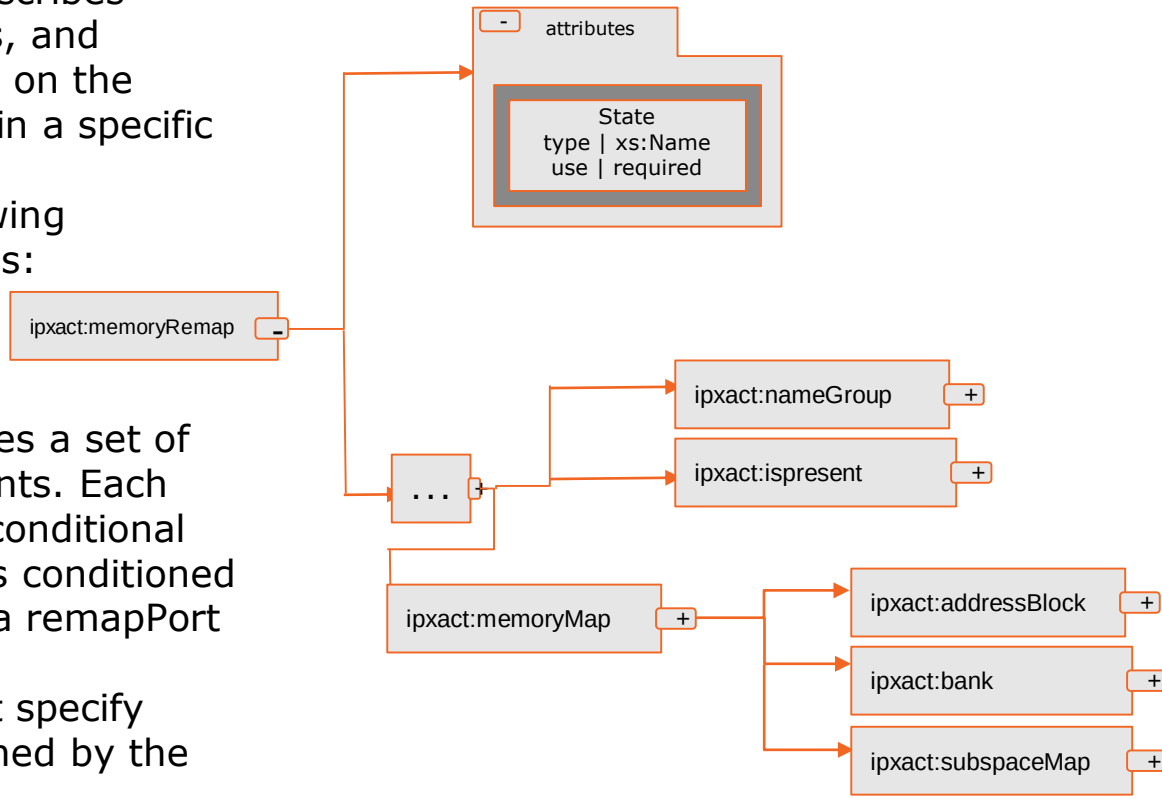
Memory Maps

- IPXACT memoryMaps contains multiple memoryMap and each memoryMap can be referred by a busInterface
- memoryMap contain multiple addressBlock and memoryRemap
- memoryRemap element describes additional address blocks, banks, and subspace maps of a slave bus interface in a specific remap state. Its like multiple memoryMaps sharing same memory location based on some signals
- Remap state is dependent on signal value

```
<ipxact:memoryMaps>
    <ipxact:memoryMap>
    <ipxact:name>rst_examplemap</ipxact:name>
    <ipxact:adressBlock>
        .   .   .   .   .
    </ipxact:adressBlock>
    <ipxact:adressUnitBits>8</ipxact:adressUnitBits>
    <ipxact:vendorExtensions>
        .   .   .   .   .
    </ipxact:vendorExtensions>
    </ipxact:memoryMap>
    <ipxact:memoryMap>
        .   .   .   .   .
    </ipxact:memoryMap>
</ipxact:memoryMaps>
```

Memory Remap

- The memoryRemap element describes additional addressBlocks, banks, and subspaceMaps that are mapped on the referencing slave bus interface in a specific remap state
- This element contains the following elements, attributes, and groups:
  - state
  - nameGroup
  - memoryMap
- A remapStates element describes a set of one or more remapState elements. Each remapState element defines a conditional remap state where each state is conditioned by a remap port specified with a remapPort element
- A remapState element does not specify remapping addresses. It is defined by the memoryRemap element



Memory Banks

- IPXACT memory bank element allows multiple addressBlocks, banks, or subspaceMaps to be concatenated together horizontally or vertically as a single entity

- It contains the following **mandatory** and optional elements:
  - **bankAlignment**: Organizes the bank
  - **nameGroup**: It shall be unique within the containing bank element
  - accessHandles:It specifies view-dependent naming for this bank within the corresponding view
  - **addressSpecifier**:It specifies the address
  - **bankBase**:It creates recursion, whereby banks maybe included inside banks included inside banks

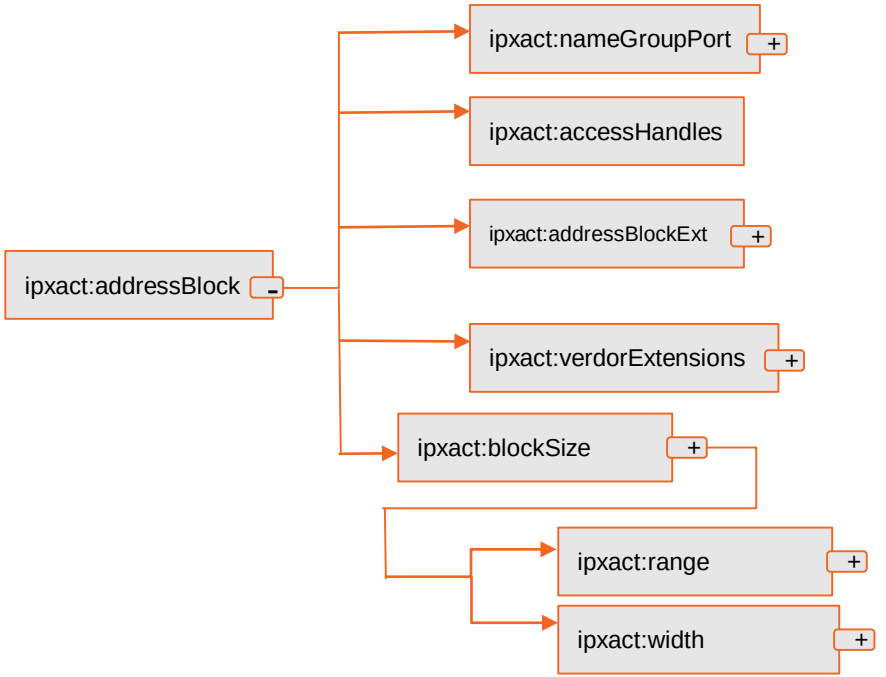
```
<ipxact:bank bankAlignment="serial">
    <ipxact:name>SerialBank</ipxact:name>
    <ipxact:baseAddress>0x1000</ipxact:baseAddress>
    <ipxact:addressBlock>
        <ipxact:name>ram0</ipxact:name>
        <ipxact:range>0x1000</ipxact:range>
        <ipxact:width>32</ipxact:width>
    </ipxact:addressBlock>
    <ipxact:addressBlock>
        <ipxact:name>ram1</ipxact:name>
        <ipxact:range>0x1000</ipxact:range>
        <ipxact:width>32</ipxact:width>
        Copyright © 2014 IEEE. All rights reserved. 485
        IEEE
        PACKAGING, INTEGRATING, AND REUSING IP WITHIN TOOL FLOWS Std 1685-2014
    </ipxact:addressBlock>
</ipxact:bank>
```

Banked address block

- The addressBlock element inside a bank element describes a single, contiguous block of memory that is part of a bank

- addressBlock contains the following mandatory and optional elements:

  - **nameGroup**: The name element shall be unique within the containing addressBlock element.
  - accessHandles: It specifies view-dependent naming for this address block within the corresponding RTL
  - **blockSize**: It includes the range and width
  - **addressBlockExtensions**: It contains optional elements commonly added to various types of address blocks in a memory map
  - vendorExtensions: It adds any extra vendor-specific data related to the address block



Banked bank

- The bank element within another bank element allows multiple address blocks, banks, or subspaceMaps to be to be concatenated together horizontally or vertically as a single entity

- The nested bank element, which can appear in another bank element is of type bankBankType. It contains the following attributes and elements:
  - bankAlignment
  - nameGroup
  - accessHandles
  - bankBase

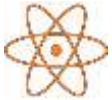
- A banked bank is similar to a bank in a memory map the only difference is there is no baseAddress element in a bank of type bankedBankType

```
<xs:complexType name="bankedBankType">
    <xs:annotation>
        <xs:documentation>
            Banks nested inside a bank do not specify address
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:group ref="ipxact:nameGroup"/>
        <xs:element name="accessHandles" minOccurs="0">
            <xs:complexType>
                <xs:sequence minOccurs="1" maxOccurs="unbounded">
                    <xs:element name="accessHandle" type="ipxact:simpleAccessHandle"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:group ref="ipxact:bankBase"/>
    </xs:sequence>
    <xs:attribute name="bankAlignment" type="ipxact:bankAlignmentType" use="required"/>
    <xs:attributeGroup ref="ipxact:id.att"/>
</xs:complexType>
```

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

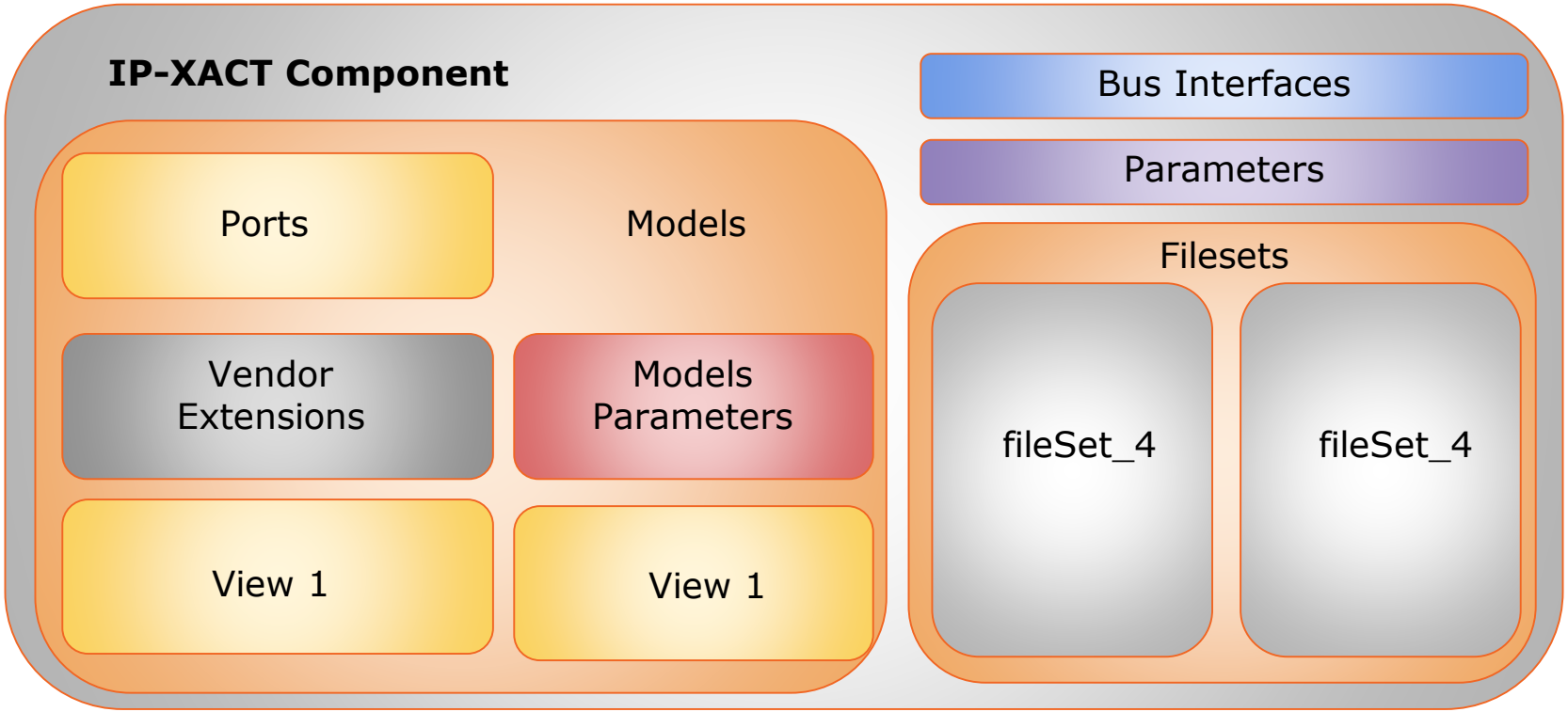
IP-XACT Model

Model

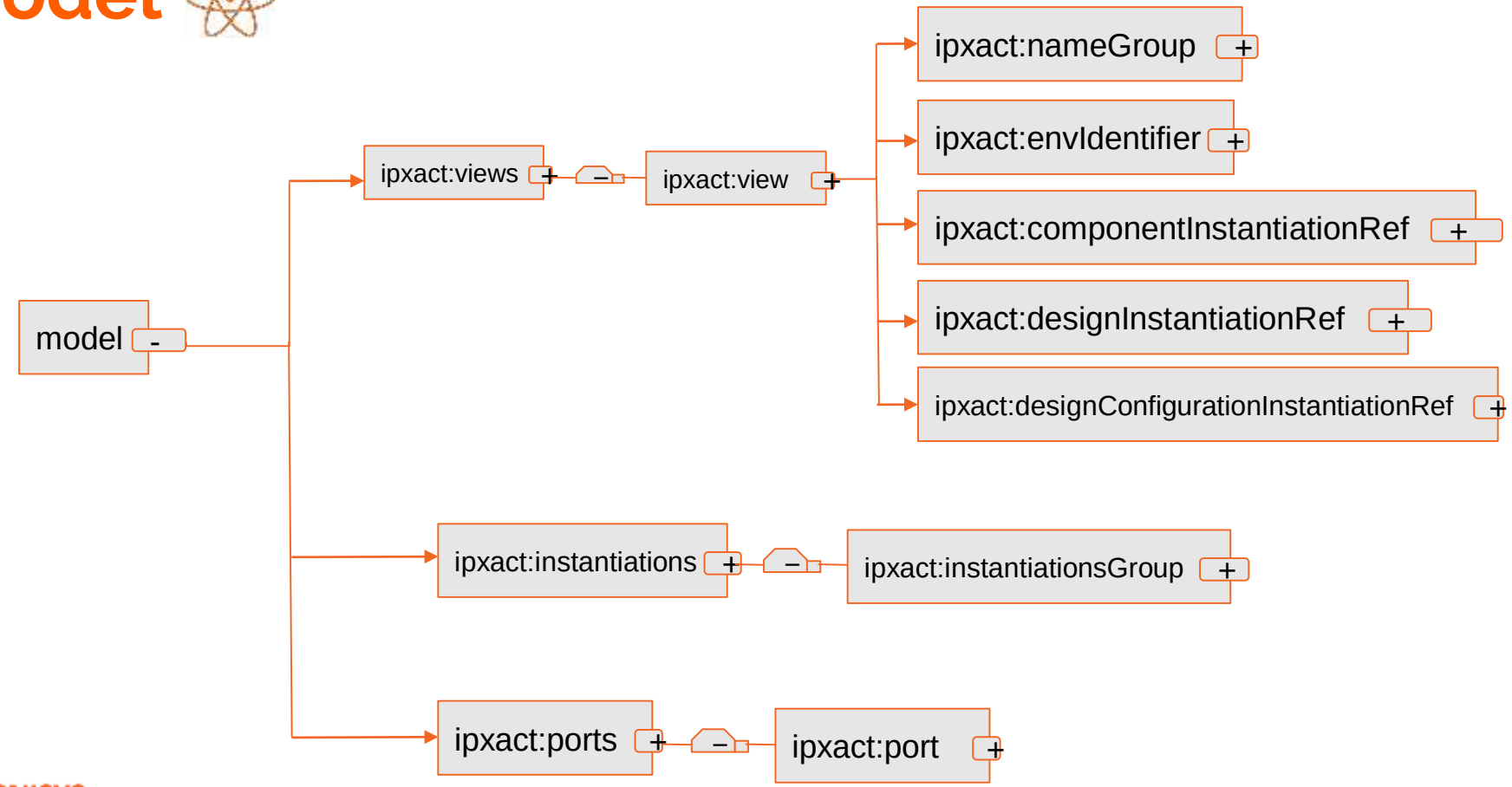


- The model element describes the views, instantiations, and ports of a component
- A model element may contain the following **mandatory** and optional elements:
  - views: A views element describes an unbounded set of view elements. Each view element specifies a representation level of a component. It further contains the following elements:
    - **nameGroup**: The name elements shall be unique within the containing views element.
    - envIdentifier: It designates and qualifies information about how this model view is deployed in a particular tool environment
    - componentInstantiationRef: It specifies a name that references a component inside the instantiations element
    - designInstantiationRef: It specifies a name that references a design inside the instantiations element
    - designConfigurationInstantiationRef: It specifies a name that references a design configuration inside the instantiations element
  - instantiations: It specifies the component, design, and design configuration instantiations for all views
  - ports: It contains the list of ports for this object. A ports is an external connection from the object

Model Structure



Model



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

IP-XACT Design

Design

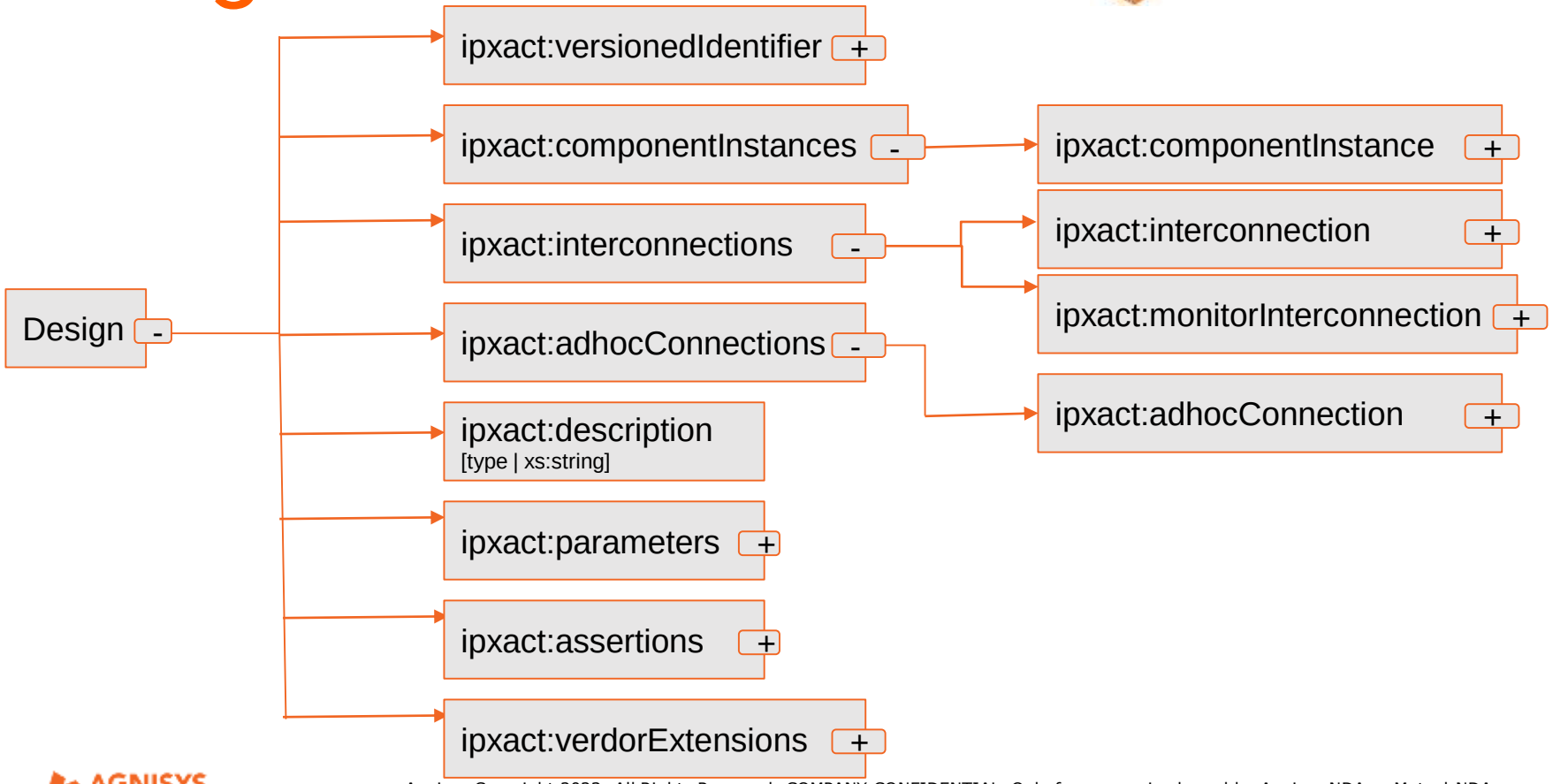
- An IP-XACT design is the central placeholder for the assembly of component objects meta-data which describes a list of components referenced by this description, their configuration, and their interconnections to each other

- A design can become a hierarchical component and it must be referenced by an IP-XACT Component XML file

- Purpose: To describe the structure of an hierarchical IP and enable generation of views related to logical interconnect and physical interconnect

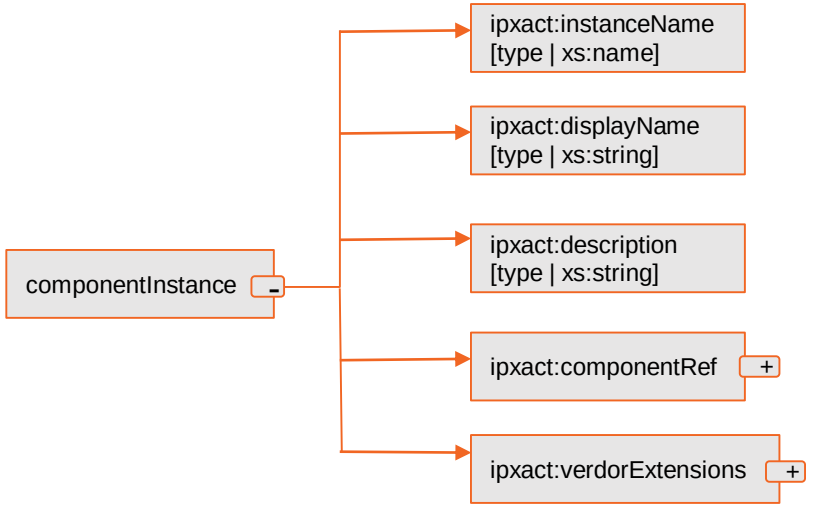
```
<ipxact:design xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:soce="http://www.agnisys.com/">
    <ipxact:vendor>Agnisys</ipxact:vendor>
    <ipxact:library>mixed_signal</ipxact:library>
    <ipxact:name>soc_top_design</ipxact:name>
    <ipxact:version>1.0</ipxact:version>
    <ipxact:componentInstances>
        <ipxact:componentInstance>
            <ipxact:instanceName>inst1</ipxact:instanceName>
            <ipxact:componentRef vendor="Agnisys" library="mixed_signal" name="block_1" [1 line]
        </ipxact:componentInstance>
        <ipxact:componentInstance> [4 lines]
    </ipxact:componentInstances>
    <ipxact:interconnections/>
</ipxact:design>
```

Design Element Information



Design: Component Instances

- The componentInstance element documents the existence of a component in a design
- This element contains the following **mandatory** and optional elements:
  - **instanceName**: Assigns a unique name for this instance of the component in this design
  - displayName: Allows a short descriptive text to be associated with the instance
  - description: Allows a textual description of the instance
  - **componentRef**: A reference to a component description
  - vendorExtensions: Adds any extra vendor-specific data related to the design

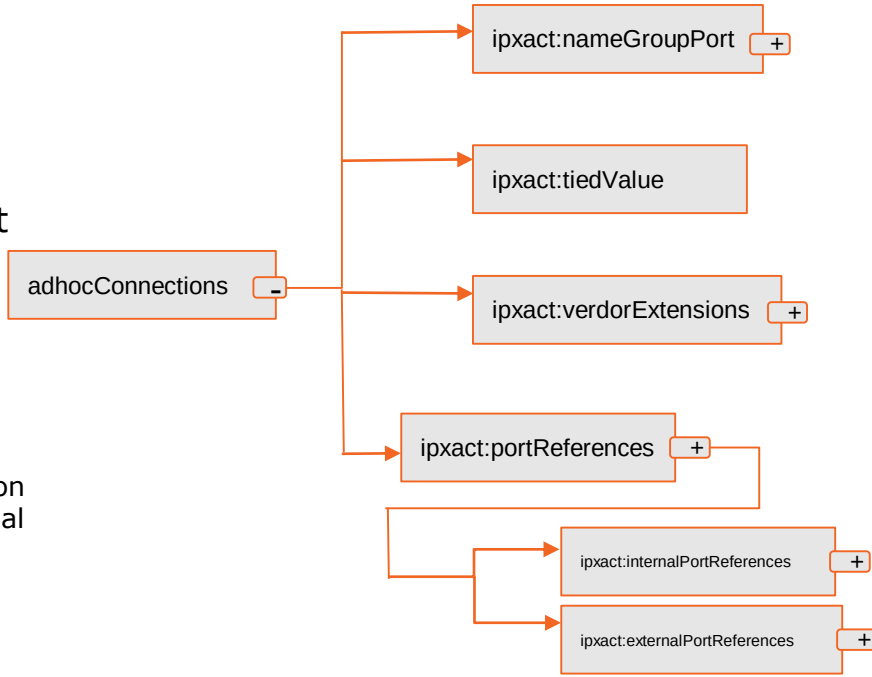


Design: ad hoc connections

- An adHocConnection specifies connections b/w component instance ports and/or b/w component instance ports and ports of the encompassing component

- Each adHocConnection shall contain at least one port reference, i.e., internal or external

- It contains the following **mandatory** and optional elements:
  - **nameGroupPort**: The name elements shall be unique within the containing adHocConnections element.
  - tiedValue: It specifies a fixed logic value for this connection
  - **portReferences**: It specifies a list on internal and external port references for this adHocConnection
    - internalPortReference: It references the port of a component instance
    - externalPortReference: It references a port of the encompassing component where this design is referred
  - vendorExtensions: It adds any extra vendor-specific data related to the design



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

IP-XACT Interface Definition

Protocol Descriptions

Bus Definition & Abstraction Definition

- Bus and abstraction definitions are referenced in component bus interfaces to indicate which interface uses which protocol and which component ports implement that protocol
- Two bus interfaces can be connected if and only if they reference the same bus definition
- Purpose: To describe aspects of an hardware communication protocol

```
<ipxact:busDefinition xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.accellera.org/
    XMLSchema/IPXACT/168
    <ipxact:vendor>accell
    <ipxact:library>i2s</
    <ipxact:name>I2S</ipx
    <ipxact:version>1.1</
    <ipxact:directConnect
    <ipxact:isAddressable
    <ipxact:description>I
</ipxact:busDefinition>
```

```
<ipxact:abstractionDefinition xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.accellera.org/
    XMLSchema/IPXACT/1685-2014 http://www.accellera.org/XMLSchema/IPXACT/1685-2014/index.xsd">
    <ipxact:vendor>accellera.org</ipxact:vendor>
    <ipxact:library>i2s</ipxact:library>
    <ipxact:name>I2S_rtl</ipxact:name>
    <ipxact:version>1.1</ipxact:version>
    <ipxact:busType vendor="accellera.org" library="i2s" name="I2S" version="1.1"/>
    <ipxact:ports> [78 lines]
</ipxact:abstractionDefinition>
```

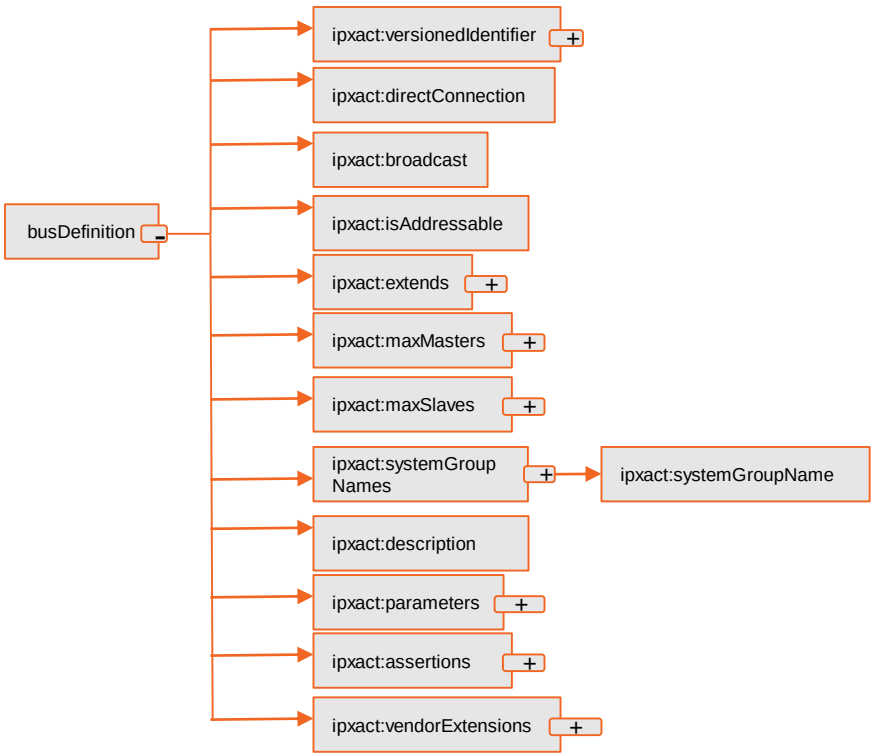
Bus Definition

- The top-level busDefinition element contains the high-level attributes of the interface, including items such as the connection method and indication of addressing

- It contains the following **mandatory** and optional elements:

  - **versionedIdentifier** - Provides a unique identifier
  - **directConnection** - Specifies what connections are allowed
  - broadcast - It means bus interfaces using this definition support one-to-many interface connections
  - **isAddressable** - Specifies the bus has addressing information
  - extends - Specifies whether this definition is an extension from another bus definition
  - maxMasters - Specifies the maximum number of masters that can appear in a channel element containing bus interfaces
  - maxSlaves - Specifies the maximum number of slaves that can appear in a channel element containing bus interfaces
  - systemGroupNames - It defines the possible group names to be used under an onSystem element in an abstraction definition
  - description - A textual description of the interface
  - parameters - It describes any parameter that can be used to configure or hold information
  - assertions - It contains a list of expressions defining the allowed parameter values
  - vendorExtensions - It contains any extra vendor-specific data related to the interface

Bus Definition

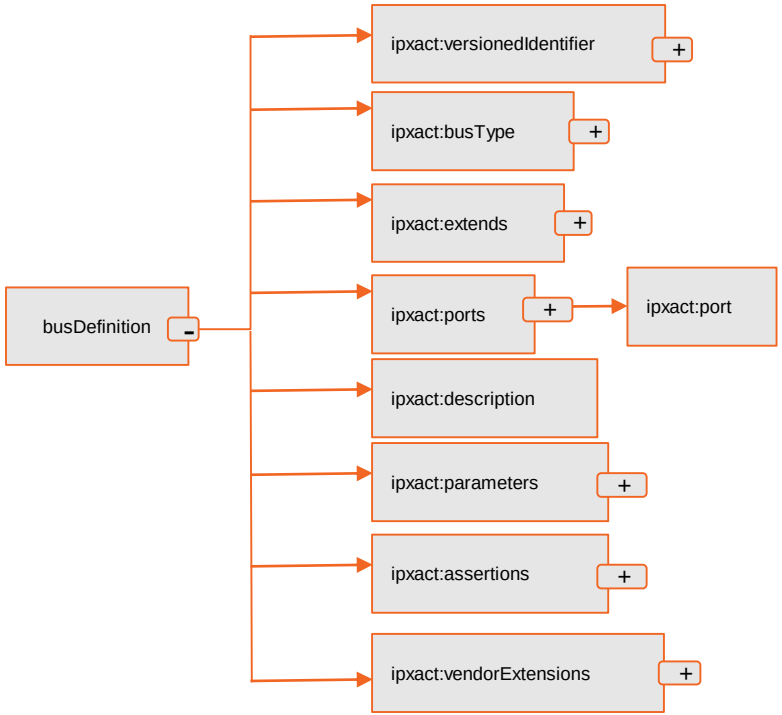



```
<ipxact:busDefinition xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/2.0"
    xsi:schemaLocation="http://www.accellera.org/XMLSchema/IPXACT/2.0/
    index.xsd">
    <ipxact:vendor>accellera.org</ipxact:vendor>
    <ipxact:library>Sample</ipxact:library>
    <ipxact:name>SampleBusDefinitionExtension</ipxact:name>
    <ipxact:version>1.0</ipxact:version>
    <ipxact:directConnection>true</ipxact:directConnection>
    <ipxact:broadcast>true</ipxact:broadcast>
    <ipxact:isAddressable>false</ipxact:isAddressable>
    <ipxact:extends vendor="accellera.org" library="Sample"
        name="SampleBusDefinitionBase" version="1.0"/>
    <ipxact:maxMasters>1</ipxact:maxMasters>
    <ipxact:maxSlaves>16</ipxact:maxSlaves>
    <ipxact:systemGroupNames>
        <ipxact:systemGroupName>SystemSignals</ipxact:systemGroupName>
    </ipxact:systemGroupNames>
    <ipxact:description>
        Example bus definition used in the IP-XACT standard.
    </ipxact:description>
</ipxact:busDefinition>
```

Abstraction Definition ↗

- The abstractionDefinition element contains a list of logical ports that define a representation of the bus type to which it refers
- This is a list of logical ports that may appear on a bus interface for that bus type
- Multiple abstractions definitions can be associated with a single bus definition
- It contains the following elements and attributes:
    - **versionedIdentifier** - It provides a unique identifier
    - **busType** - It specifies the bus definition that this abstraction defines
    - extends - It specifies whether this definition is an extension from another abstraction definition
    - **ports** - It is a list of logical ports
    - description - It allows a textual description of the interface
    - parameters - It describes any parameter that can be used to configure or hold information
    - assertions - It contains a list of expressions defining the allowed parameter values
    - vendorExtensions - It contains any extra vendor-specific data

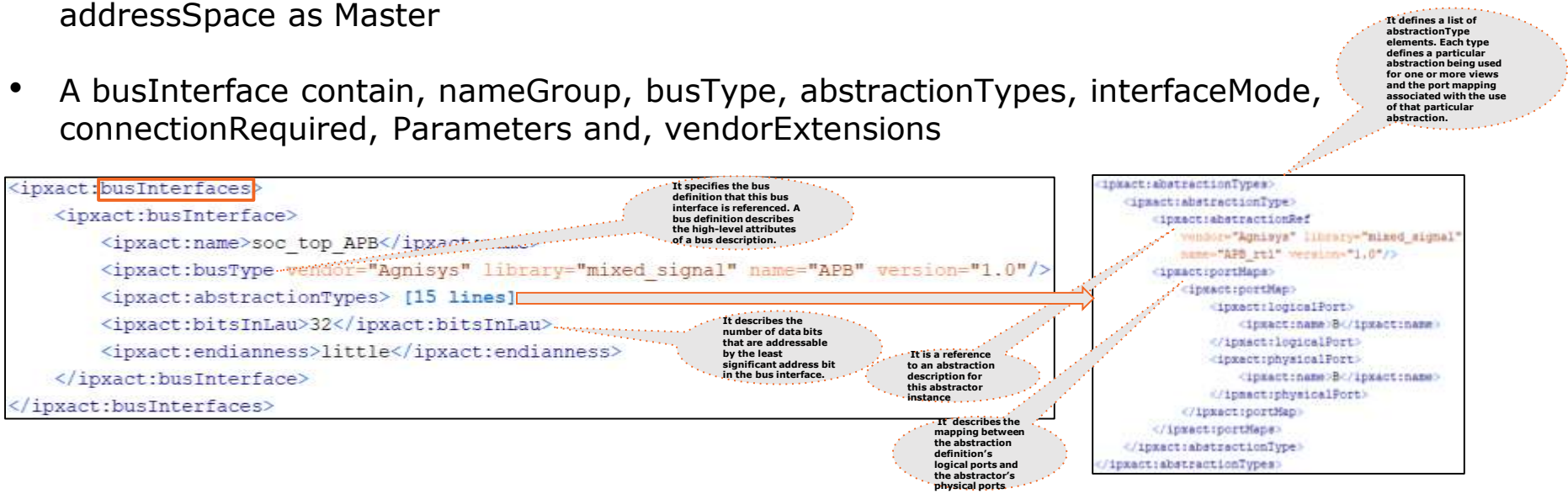
Abstraction Definition




```
<ipxact:abstractionDefinition xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
    xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/2.0"
    xsi:schemaLocation="http://www.accellera.org/XMLSchema/IPXACT/2.0/
    index.xsd">
    <ipxact:vendor>accellera.org</ipxact:vendor>
    <ipxact:library>Sample</ipxact:library>
    <ipxact:name>SampleAbstractionDefinition_TLM</ipxact:name>
    <ipxact:version>1.0</ipxact:version>
    <ipxact:busType vendor="accellera.org" library="Sample"
        name="SampleBusDefinitionExtension" version="1.0"/>
    <ipxact:ports> [33 lines]
    <ipxact:description>Example TLM abstraction definition used in the IP-XACT [1 line]
</ipxact:abstractionDefinition>
```

Bus Interface

- A busInterface is a grouping of ports related to a function, typically a bus, defined by a bus definition and abstraction definition
- Each Interface in busInterface is assigned with some memoryMap as slave or addressSpace as Master
- A busInterface contain, nameGroup, busType, abstractionTypes, interfaceMode, connectionRequired, Parameters and, vendorExtensions



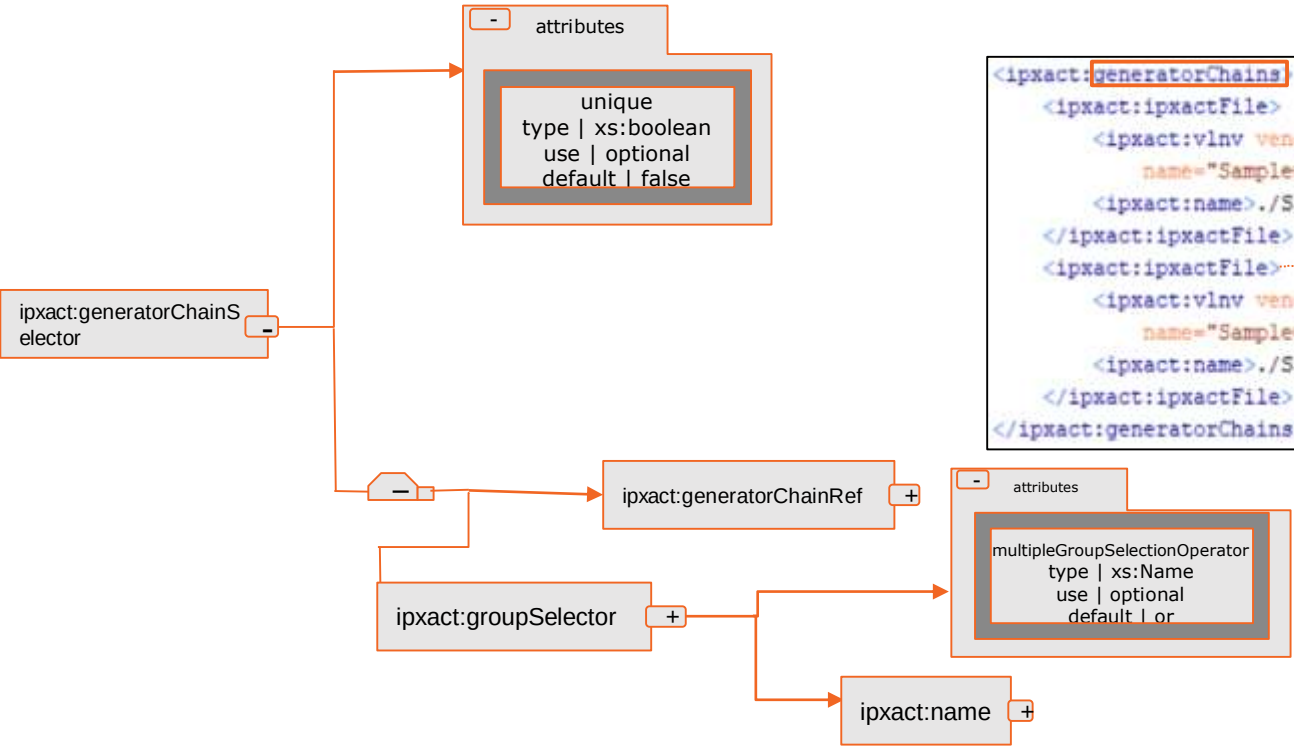
AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

IP-XACT Generator Chain

Generator Chain

- A generator chain is an ordered sequence of named tasks
- Each named task can be represented as a single generator or as another generator chain.
- The generatorChain element contains the following **mandatory** and optional elements:
  - **versionedIdentifier**: It provides a unique identifier; it consists of four sub elements for a top-level IP-XACT element.
  - **generatorChainSelector** : It contains one or more of the following subelements:
    - *generatorChainSelector*, *componentGeneratorSelector*, and *generator*
  - chainGroup: It is an unbounded list of names to which this chain belongs
  - displayName: It allows a short descriptive text to be associated with the generator chain
  - description: It allows a textual description of the generator chain
  - choices: It specifies multiple enumerated lists, which are referenced by other sections of this generator chain description
  - parameters: It describes any parameter that can be used to configure or hold information related to this generator chain
  - assertions: It contains a list of expressions defining the allowed parameter values
  - vendorExtensions: It contains any extra vendor-specific data related to the generatorChain

Generator Chain




```
<ipxact:generatorChains>
    <ipxact:ipxactFile>
        <ipxact:vlnv vendor="accellera.org" library="Sample"
            name="SampleGeneratorChain" version="1.0"/>
        <ipxact:name>./SampleGeneratorChain.xml</ipxact:name>
    </ipxact:ipxactFile>
    <ipxact:ipxactFile>
        <ipxact:vlnv vendor="accellera.org" library="Sample"
            name="SampleGeneratorChainForReference" version="1.0"/>
        <ipxact:name>./SampleGeneratorChainForReference.xml</ipxact:name>
    </ipxact:ipxactFile>
</ipxact:generatorChains>
```

It describes the number of data bits that are addressable by the least significant address bit in the bus interface.

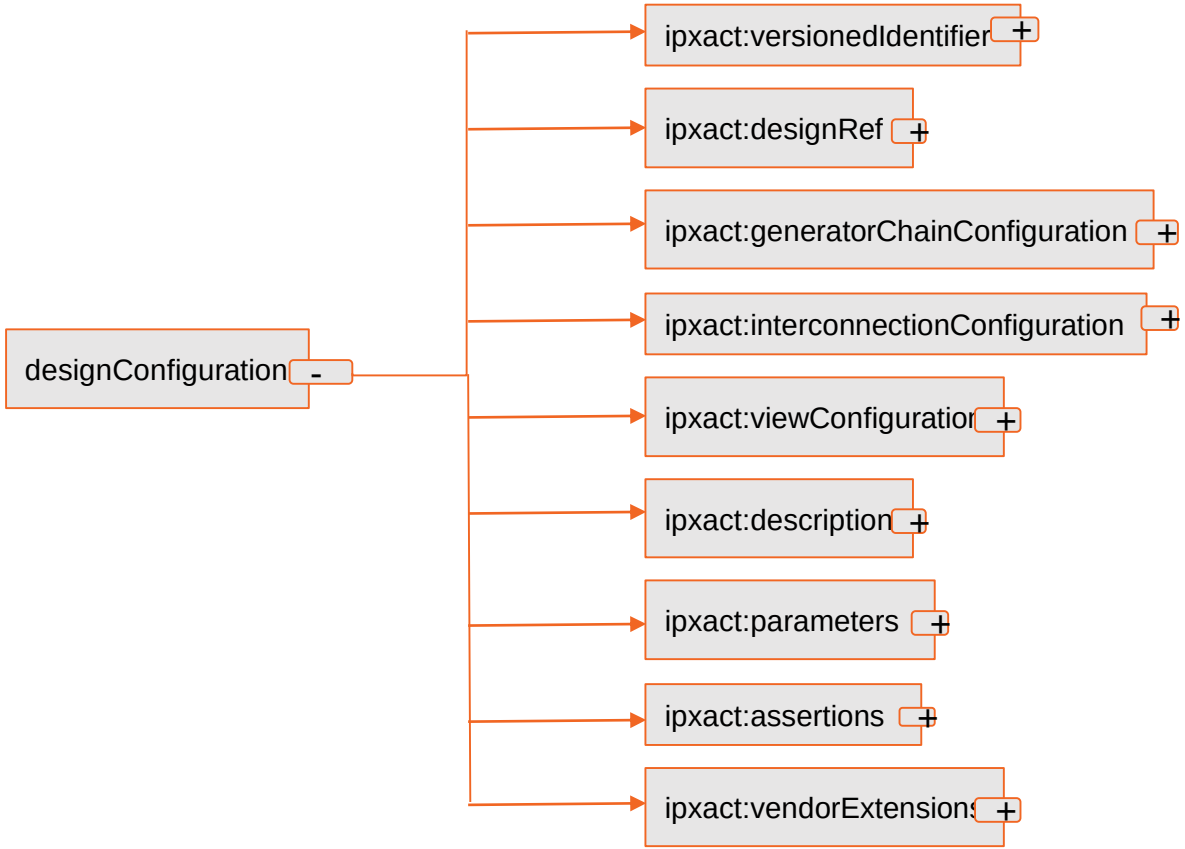
AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

IP-XACT Design Configuration

Design Configuration

- An IP-XACT design configuration is a placeholder for additional configuration information of a design or generator chain description
- The design configuration description contains the following configuration information:
  - Configuration values for parameters defined in generators within generator chains
  - Active view or current view selection for instances in the design description along with configuration values for view parameters within the component instances
  - Configuration information for interconnections between the same bus types with differing abstraction types
- A design configuration applies to a single design, but a design may have multiple design configuration descriptions

Design Configuration

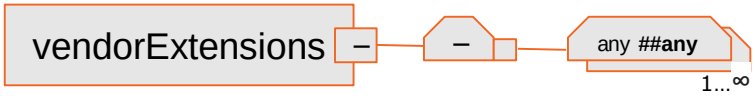


Design Configuration

- The designConfiguration element contains the following **mandatory** and optional elements:
  - **versionedIdentifier**: It provides a unique identifier; it consists of four sub elements for a top-level IP-XACT element
  - designRef: It specifies the design description for this design configuration
  - generatorChainConfiguration: It documents a generator chain VLNV and an unbounded list of configuration values for parameters within the referenced generator chain
  - interconnectionConfiguration: It is an unbounded list of information associated with interface interconnections
  - viewConfiguration: It defines the selected view and view configuration for an instance of the design
  - description: It allows a textual description of the design configuration
  - parameters: It describes any parameter that can be used to configure or hold information related to this design configuration
  - assertions: It contains a list of expressions defining the allowed parameter values
  - vendorExtensions: It adds any extra vendor-specific data related to the design configuration

Vendor Extensions

- The vendorExtensions element is a place in the description in which any vendor-specific information can be stored
- The vendorExtensions element allows any well-formed description




```
<ipxact:vendorExtensions>
    <any_example:extraElement xmlns:any_example="http://www.nosuchURL.org/Schema">
        <any_example:anotherElement any_example:attributeName="XYZ"/>
    </any_example:extraElement>
</ipxact:vendorExtensions>
```

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

IP-XACT Support in IDesignSpec

Support of IP-XACT Properties in IDS

- Properties are handled using vendor extensions
- Vendor extensions are well-defined locations in IP-XACT documents where information can be added that cannot be described in the IP-XACT schema

```
<ipxact:vendorExtensions>
<ids_properties>
    <access>read-write</access>
    <abs_hdl_path>testing_unit_ids.addrblk_name2_reg_0</abs_hdl_path>
    <address>0x000000</address>
    <hdl_path>addrblk_name2_reg_0</hdl_path>
</ids_properties>
</ipxact:vendorExtensions>
```

IP-XACT Properties in IDS

Properties	Description
vendor	Specifies the name of the vendor of the IP-XACT model
library	Specifies the name of the library of the IP-XACT model
version	Specifies the version number of the IP-XACT model
ipxact_topdesign	Specifies the top design file name
ipxact_buscomp	Specifies the bus component file name
ipxact_ignore	Removes the bus/ports from bus component node.
ipxact_compact	Removes the section hierarchy of memoryMap and addressBlock
-addr_sort	Sorts registers in ascending order

Properties	Description
ipxact_exclude_param	Removes parameter while importing the IPXACT file in IDS.
ipxact_decrease_offset	Decreases the offset value in ipxact file and is referred by in any input type
ipxact_exclude_vendor extensions	Used for excluding "spirit:vendorExtensions" nodes from the IP-XACT output.
-ipxact_inp_fast	Generates output through java
-ipxact_comp	Output is generated as per the specified hierarchy
-if_ipxact	Generates separate files for each block
xml_addr_mux	Multiplies each register offset with the property value
ipxact_rm_alt_reg	For removing the alternate registers from IP-XACT

IP-XACT Properties & IDS Properties

IDesignSpec™	IP-XACT 2014	IP-XACT 2009
RegGroup	<ipxact:registerFile>	<spirit:registerFile>
RegGroup size	<ipxact:range>	<spirit:range>
RegGroup offset	<ipxact:addressOffset>	<spirit:addressOffset>
Register	<ipxact:register>	<spirit:register>
Register Name	<ipxact:name>	<spirit:name>
Register Offset	<ipxact:addressOffset>	<spirit:addressOffset>
Default size	<ipxact:size>	<spirit:size>
Enum	<ipxact:enumeratedValues>	<spirit:enumeratedValues>
HW Access (wo or rw)	<ipxact:volatile>	<spirit:volatile>
Default	<ipxact:reset>	<spirit:reset>
Description	<ipxact:description>	<spirit:description>
Field	<ipxact:field>	<spirit:field>
LSB bit of field	<ipxact:bitOffset>	<spirit:bitOffset>
Bit width (MSB-LSB)	<ipxact:bitWidth>	<spirit:bitWidth>
SW Access	<ipxact:access>	<spirit:access>
Repeat	<ipxact:dim>	<spirit:dim>

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Advanced Topics

File Sets

- Multiple file sets can be described using the container element fileSet. Each fileSet element has a name to identify it and a list of files

- Each file has a name element whose value is a path to a source file which can be a relative path w.r.t the location of the IP-XACT file or an absolute path possibly using an environment variable

```
<ipxact:fileSets>
    <ipxact:fileSet>
        <ipxact:name>fs-rtl</ipxact:name>
        <ipxact:file>
            <ipxact:name>../RTL/transmitter_is_master_rtl.v</ipxact:name>
            <ipxact:fileType>verilogSource</ipxact:fileType>
            <ipxact:logicalName>transmitter_is_masterlib</ipxact:logicalName>
        </ipxact:file>
    </ipxact:fileSet>
</ipxact:fileSets>
```

Enumerated Values

- The enumeratedValues element is a container element for enumeratedValue elements

- An enumeratedValue describes a value of the containing field. It can have an attribute usage describing the usage of the enumeratedValue which can take the values read, write, and read-write
- Furthermore, an enumeratedValue has a name to identify the element. It can have a description to provide a human-readable description

```
<ipxact:enumeratedValues>
   <ipxact:enumeratedValue usage="read">
      <ipxact:name>EMPTY</ipxact:name>
      <ipxact:description>RX-FIFO empty</ipxact:description>
      <ipxact:value>0</ipxact:value>
   </ipxact:enumeratedValue>
   <ipxact:enumeratedValue usage="read">
      <ipxact:name>NOT_EMPTY</ipxact:name>
      <ipxact:description>RX-FIFO not empty.</ipxact:description>
      <ipxact:value>1</ipxact:value>
   </ipxact:enumeratedValue>
</ipxact:enumeratedValues>
```

Comparison between SystemRDL & IP-XACT

SystemRDL	IP-XACT 2014
addrmap	<ipxact:memoryMap>
addrmap	<ipxact:addressBlock>
<block name> <block_instance> @offset	<ipxact:addressOffset>
external	Handled with the help of vendor extension
regwidth	<ipxact:width>
regfile	<ipxact:registerFile>
<regroup name> <regroup_instance>@offset	<ipxact:addressOffset>
reg	<ipxact:register>
reg <register_name>	<ipxact:name>
<reg name> <reg_instance>@offset	<ipxact:addressOffset>
hw=wo/rw	<ipxact:volatile>
default	<ipxact:reset>
desc	<ipxact:description>
field	<ipxact:field>
[Lsb]	<ipxact:bitOffset>
[Msb : Lsb]	<ipxact:bitWidth>
sw	<ipxact:access>
[<repeat_value>]	<ipxact:dim>
User Defined Properties	Vendor Extensions

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

IP-XACT Tight Interface Generator

Introduction to TGI

- The Tight Generator Interface provides an API to query, modify, create, and delete IP-XACT documents residing in an IP-XACT compliant design tool
- It uses the concept of handles to objects. Handles are called identifiers (IDs) while, Objects are entities that can be described in IP-XACT documents
- There are two classes of identifiers:
  - Unconfigured IDs: Provides access to an object type
  - Configured IDs: Provides access to an object instance
- TGI can also be used to develop generators to create IP-XACT documents from proprietary views

TGI APIs

getDesignAdHocConnectionIDs	getAddressBlockRegisterID	getRegisterFieldTestContraint	getRegisterFieldAccess	getDisplayName
getDesignComponentInstanceIDs	getMemoryMapElementIDs	getRegisterFieldTypeIdentifier	getRegisterFieldBitOffset	getName
getDesignInterconnectionIDs	getParameterIDs	getRegisterFieldVolatility	getRegisterFieldBitWidth	setDescription
setAdHocConnectionTiedValue	getPortDirection	getResetMask	getBridgeMasterID	setDisplayName
getFileSetFileIDs	getEnumeratedValueUsage	getResetMaskExpression	getComponentBusInterfaceIDs	getRegisterAccess
getComponentIDs	getEnumeratedValueValue	getResetTypeRef	getComponentFileSetIDs	getRegisterAddressOffset
getDesignID	getRegisterFieldReserved	getResetValue	getComponentMemoryMapIDs	getRegisterDimensions
getDesignIDs	getRegisterFieldResetIDs	getResetValueExpression	getComponentPortIDs	getRegisterFieldIDs
getGeneratorContextComponentInstanceID	getRegisterFieldTestable	getAbstractionDefinitionBusTyp	getComponentViewIDs	getRegisterResetMask
setConfigurableElementValue	getRegisterFieldReadAction	getAbstractionDefinitionExtend	getDescription	getRegisterResetValue
createDesign	getComponentGeneratorIDs	getRegisterSize	getAbstractionDefinitionPortIDs	getAbstractionDefPortIsCl
getAdHocConnectionTiedValue	getComponentRemapStateIDs	getBusInterfaceSlaveBridgeIDs	getAbstractionDefPortDefaultValu	getAbstractionDefPortIsD
getComponentInstanceName	getConfigurableElementValue	getComponentChoiceIDs	getAbstractionDefPortIsAddress	getAbstractionDefPortIsRe

TGI APIs

getAbstractionDefPortMirroredModeConstraintIDs	getAbstractionDefPortModeProtocolPayloadType	addRegisterField	addBusInterfaceSlaveBridge
getAbstractionDefPortModeBusWidth	getAbstractionDefPortModeProtocolType	removeComponentBusInterface	addComponentChoice
getAbstractionDefPortModeConstraintIDs	getAbstractionDefPortModeProtocolTypeCustom	removeComponentPort	addComponentRemapState
getAbstractionDefPortModeDirection	getAbstractionDefPortModeWidth	removeComponentView	addMemoryMapAddressBlock
getAbstractionDefPortModeGroup	getAbstractionDefPortRequiresDriver	setRegisterAccess	removeAttribute
addAbstractionDefPortMirroredModeTimingConstraint	getAbstractionDefPortRequiresDriverType	setRegisterFieldAccess	removeComponentChoice
getAbstractionDefPortModeInitiative	getAbstractionDefPortStyle	setRegisterDimensions	setBusInterfaceMasterAddressSpaceName
addAbstractionDefPortMirroredModeLoadConstraint	addComponentAddressSpace	setRegisterFieldModifiedWriteValue	setBusInterfaceMasterBaseAddress
getAbstractionDefPortModeKindCustom	addComponentBusInterface	setRegisterFieldReadAction	setBusInterfaceSlaveMemoryMapName
addAbstractionDefPortMirroredModeDriveConstraint	addComponentFileSet	setRegisterResetMask	addAbstractionDefinitionPort
getAbstractionDefPortModeProtocolPayloadExtension	addComponentMemoryMap	setRegisterResetValue	getAbstractionDefPortModePresence
getAbstractionDefPortModeProtocolPayloadExtensionMandatory	addComponentView	addAddressBlockRegister	getAbstractionDefPortModeKind
getAbstractionDefPortModeProtocolPayloadName	addRegisterAlternateRegister	addAttribute	getAbstractionDefPortModeIDs

TGI Use Case

Input IP-

```
<ipxact:component xmlns:xsi="http:
                  xmlns:ipxact="ht
                  xsi:schemaLocati
  <ipxact:vendor>accellera.org</ip
  <ipxact:library>Sample</ipxact:l
  <ipxact:name>SampleComponent</ip
  <ipxact:version>1.0</ipxact:vers
  <ipxact:busInterfaces> [179 lines]
  <ipxact:indirectInterfaces> [7 li
  <ipxact:channels> [10 lines]
  <ipxact:remapStates> [9 lines]
  <ipxact:addressSpaces> [6 lines]
  <ipxact:memoryMaps> [170 lines]
  <ipxact:model> [256 lines]
  <ipxact:componentGenerators> [22
  <ipxact:choices> [6 lines]
  <ipxact:fileSets> [16 lines]
  <ipxact:whiteboxElements> [7 line
  <ipxact:cpus> [5 lines]
  <ipxact:otherClockDrivers> [7 lin
  <ipxact:resetTypes> [5 lines]
  <ipxact:description>Example comp
  <ipxact:parameters> [17 lines]
  <ipxact:assertions> [6 lines]
</ipxact:component>
```

run.tcl

```
soc_read -search_path "input" -file "sample.ipxact.xml"
puts " start reading "
set getid [ tgi::getID { "accellera.org" "Sample" "SampleComponent" "1.0" } ]
puts "1.getID: $getid"
set busInterface [ tgi::getComponentBusInterfaceIDs $getid ]
puts "2.BusInterfaceID: $busInterface"
set InterfaceSlaveBridgeID [ tgi::getBusInterfaceSlaveBridgeIDs $busInterface ]
puts "3.BusInterfaceSlaveBridgeIDs: $InterfaceSlaveBridgeID"
set memMapList [  tgi::getComponentMemoryMapIDs $getid ]
puts "4.MemoryMapList: $memMapList"
set compList [tgi::getComponentIDs]
puts "5.ComponentDesignConfigurationInstantiationID: $ConfigurationInstantiationID"
foreach x $memMapList {
    puts "entering memory map"
    set memMapElements [ tgi::getMemoryMapElementIDs $x ]
    puts "16.MemoryMapElementID: $memMapElements"
    foreach y $memMapElements {
        set regList [ tgi::getAddressBlockRegisterIDs $y ]
        puts "17.registerID: $regList"
        foreach z $regList {
        set RegTypeIdentifier [ tgi::getRegisterTypeIdentifier $z ]
        puts "18.RegisterTypeIdentifier: $RegTypeIdentifier"
        set Volatility [ tgi::getRegisterVolatility $z ]
        puts "19.RegisterVolatility: $Volatility"
        puts "regname: $z"
        set regis [ tgi::getRegisterAlternateRegisterIDs $z ]
        puts "21.RegisterAlternateRegisterID: $regis "
        set RegFieldID [ tgi::getRegisterFieldIDs $z ]
        puts "20.RegisterFieldID: $RegFieldID"
        foreach q $RegFieldID {
            set ConstraintMinMax [ tgi::getRegisterFieldWriteValueConstraintMinMax
            puts "22.RegisterFieldWriteValueConstraintMinMax: $ConstraintMinMax"
            set FieldTestContraint [ tgi::getRegisterFieldTestContraint $q ]
            puts "23.RegisterFieldTestContraint: $FieldTestContraint"
        }
      }
   }
}
soc_generate -out {"v"} -dir "idsl"
```

Generated Verilog Output

```
        parameter TLMModelsAvailable = 0,
        parameter comp_dual_mode = 1,
        parameter addrBits = 32

    ){
    input slv_data,
    output mst_data,
    input slv_addr,
    output mst_addr,
    input slv_parity,
    input clk,
    input reset,
    output status,
    input anotherPort,
    input slv_transaction,
    input mst_transaction
    );
endmodule
```

```
`include "SampleComponent.v"
module top_blk #(
        parameter addr_width = 32,
        parameter bus_width = 32

    );
    wire wire_sample_inst_mst_data;
    wire wire_sample_inst_mst_addr;
    wire wire_sample_inst_status;
SampleComponent #( .TLMModelsAvailable(0),
.comp_dual_mode(1), .addrBits(32)) sample_inst(
    .slv_data(1'bx),
    .mst_data(wire_sample_inst_mst_data),
    .slv_addr(1'bx),
    .mst_addr(wire_sample_inst_mst_addr),
    .slv_parity(1'bx),
    .clk(1'bx),
    .reset(1'bx),
    .status(wire_sample_inst_status),
    .anotherPort(1'bx),
    .slv_transaction(1'bx),
    .mst_transaction(1'bx)
);
endmodule
```

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

IP-XACT Use Case Scenario

IP-XACT as an output (IDS-Integrate)

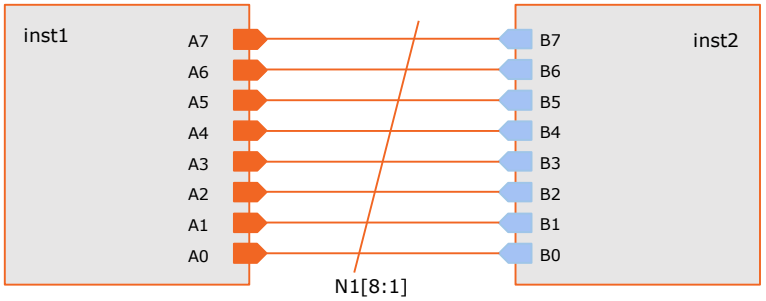
run.tcl file

```
soc_create -type block -name soc_top-top
soc_create -type block -name block_1 -port {output [7:0] A}
soc_create -type block -name block_2 -port {input [7:01 B}

soc_add -type block -parent soc_top -name block_1 -inst inst1
soc_add -type block -parent soc_top -name block_2 -inst inst2

soc_connect -source_inst inst1.A\[7:01 -dest_inst inst2.B\[7:0]

soc_generate -compact -out {v,IP-XACT} -dir "ids_c_tcl"
```



Bus net connection to single port

(Top level

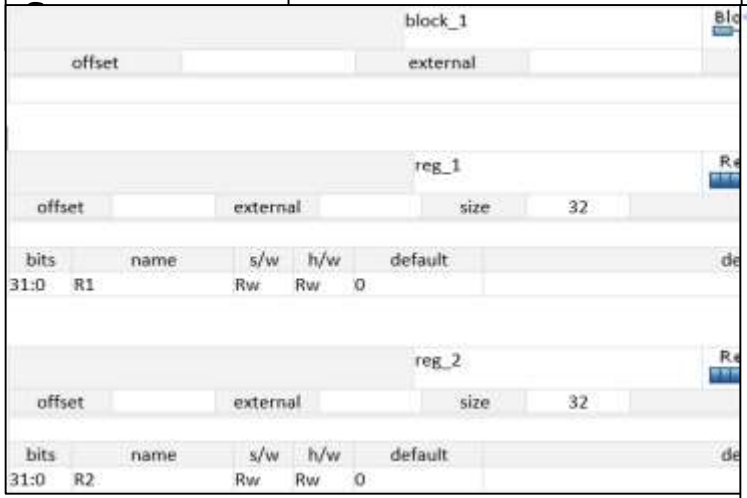
```
<ipxact:component xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:soce="http://www.agnisys.com/">
    <ipxact:vendor>Agnisys</ipxact:vendor>
    <ipxact:library>mixed_signal</ipxact:library>
    <ipxact:name>soc_top</ipxact:name>
    <ipxact:version>1.0</ipxact:version>
```

(Top level

```
<ipxact:design xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:soce="http://www.agnisys.com/">
    <ipxact:vendor>Agnisys</ipxact:vendor>
    <ipxact:library>mixed_signal</ipxact:library>
    <ipxact:name>soc_top_design</ipxact:name>
    <ipxact:version>1.0</ipxact:version>
    <ipxact:componentInstances>
        <ipxact:componentInstance>
            <ipxact:instanceName>inst1</ipxact:instanceName>
            <ipxact:componentRef vendor="Agnisys" library="mixed_signal" name="block_1"
                version="1.0"/>
        </ipxact:componentInstance>
        <ipxact:componentInstance>
            <ipxact:instanceName>inst2</ipxact:instanceName>
            <ipxact:componentRef vendor="Agnisys" library="mixed_signal" name="block_2"
                version="1.0"/>
        </ipxact:componentInstance>
    </ipxact:componentInstances>
    <ipxact:interconnections/>
</ipxact:design>
```

IP-XACT as an output (IDesignSpec)

IDS Input



IP-XACT Output

```
<ipxact:component xmlns:vendorExtensions="http://www.agnisys.com/"
                  xmlns:ids="http://www.agnisys.com/"
                  xmlns:xrsl="http://www.agnisys.com/"
                  xmlns:snps="http://www.synopsys.com/"
                  xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014">
  <ipxact:vendor>Agnisys_Inc.</ipxact:vendor>
  <ipxact:library>Agnisys_Inc.</ipxact:library>
  <ipxact:name>block1_model</ipxact:name>
  <ipxact:version>1.0</ipxact:version>
  <ipxact:busInterfaces> [123 lines]
  <ipxact:memoryMaps>
    <ipxact:memoryMap>
  <ipxact:name>block1map</ipxact:name>
    <ipxact:addressBlock>
      <ipxact:name>block1</ipxact:name>
      <ipxact:description/>
        <ipxact:baseAddress>0x0</ipxact:baseAddress>
      <ipxact:range>0x8</ipxact:range>
      <ipxact:width>32</ipxact:width>
      <ipxact:register>
        <ipxact:name>reg1</ipxact:name>
        <ipxact:addressOffset>0x0</ipxact:addressOffset>
        <ipxact:size>32</ipxact:size>
        <ipxact:volatile>true</ipxact:volatile>
        <ipxact:field> [17 lines]
        <ipxact:vendorExtensions> [11 lines]
      </ipxact:register>
      <ipxact:register>
        <ipxact:name>reg2</ipxact:name>
```

IP-XACT as an input (IDS-Integrate)

run.tcl file

```
soc_read -search_path "./input" -file "gpio.v,timer.v,ahb_master.v"
soc_read -search_path "./input" -file "gpio.v,timer.v" -inc_dir "./input" -include "apb_widget.v"
soc_read -search_path "./input" -file "basic_test.ipxact.xml"

soc_create -type block -name block1_top -top -bus apb

#soc_add -target block1_top -type block -name chipnamemodel_ids -inst chipnamemodel_ids_inst
soc_add -target block1_top -type block -name Block1_model -inst block1_model_inst -base 0x0 -size 0x

#soc_add -target block1_top -type block -name ahb_master -inst ahb_master_inst -base 0x0 -size 0x100
soc_add -target block1_top -type block -name gpio_ids -inst gpio_ids_inst -base 0x10000 -size 0x100
soc_add -target block1_top -type block -name timer_ids -inst timer_ids_inst -base 0x20000 -size 0x1
soc_add -type ports -target block1_top -bus ahb -port {input p1, input p2,output [0:2]p3, input p4 [

#soc_connect -dest block1_top.p1 -source_inst timer_ids_inst.irq
soc_connect -dest inst_gpio_ids_inst.status_fld_in -source_inst timer_ids_inst.counter_f1_r
soc_con                                            timer_ids_inst -bus apb
soc_con                                            gpio_ids_inst -bus apb
soc_con                                            block1_model_inst -bus apb
#soc_cr
soc_pro

soc_ge

#soc_gr

puts "
```

Input IP-XACT

```
<spirit:component xmlns:xsi="http://www.agnisys.com/" xmlns:xsd="http://www.synopsys.com/"
                  xmlns:xsi="http://www.agnisys.com/"
                  xmlns:spirit="http://www.spiritconsortium.org/XMLSchema/SPIRIT/1685-2014">
  <spirit:vendor>Agnisys_Inc.</spirit:vendor>
  <spirit:library>Agnisys_Inc.</spirit:library>
  <spirit:name>Block1_model</spirit:name>
  <spirit:version>1.0</spirit:version>
  <spirit:busInterfaces>
    <spirit:busInterface>
      <spirit:name>default_map</spirit:name>
      <spirit:busType spirit:vendor="AGNISYS" spirit:library="AGNISYS" spirit:name="APB" [1 line]
      <spirit:abstractionType spirit:vendor="AGNISYS" spirit:library="AGNISYS" spirit:name="APB_rtl" [1 line]
      <spirit:slave> [0 lines]
      <spirit:portMaps>
        <spirit:portMap> [7 lines]
        <spirit:portMap> [7 lines]
        <spirit:portMap> [7 lines]
        . . . . . .
        . . . . . .
      </spirit:portMaps>
      <spirit:vendorExtensions> [3 lines]
    </spirit:busInterface>
  </spirit:busInterfaces>
  <spirit:memoryMaps> [81 lines]
  <spirit:model>
    <spirit:ports>
      <spirit:port> [8 lines]
      <spirit:port> [8 lines]
      . . . . . .
      . . . . . .
      <spirit:port> [10 lines]
      <spirit:port> [8 lines]
    </spirit:ports>
    <spirit:modelParameters> [9 lines]
  </spirit:model>
  <spirit:vendorExtensions> [4 lines]
</spirit:component>
```

Verilog output

```
include "../input/gpio.v"
include "ids_block1_top_aggregation_APB_inst_apb_aggregation.v"
include "../input/timer.v"

module block1_top #(
        parameter addr_width = 'h20,
        parameter bus_width = 'h32,
        parameter block1_top_offset = 'h0,
        parameter gpio_ids_address_width = 'hc,
        parameter gpio_ids_offset = 'h0,
        parameter timer_ids_address_width = 'hd,
        parameter timer_ids_offset = 'h0

    ) (. . . . . .
       . . . . . .
       . . . . . .)

);

ids_block1_top_aggregation_APB_inst_apb_aggregation #( .addr_width('h20),
.bus_width('h32), .block1_top_aggregation_APB_inst_offset(block1_top_offset),
.timer_ids_inst_addr_width(timer_ids_address_width),
.gpio_ids_inst_addr_width(gpio_ids_address_width)) block1_top_aggregation_APB_inst(
    .pclk(pclk),
    .presetn(presetn),
    . . . . . .
    . . . . . .
    .gpio_ids_inst_ids_prdata(wire_gpio_ids_inst_prdata_block1_top_aggregation_APB_instgpio_ids_inst_ids_prdata),
    .gpio_ids_inst_ids_pslverr(wire_gpio_ids_inst_pslverr_block1_top_aggregation_APB_instgpio_ids_inst_ids_pslverr)

);
timer_ids #( .NUM_TIMER('h1), .NUM_SRC('h8), .PRESCALER_WIDTH('h20), .COUNTER_WIDTH('h20),
.bus_width('h32), .addr_width(timer_ids_address_width), .block_offset(timer_ids_offset),
.counter_f1_incr_val('h0), .counter_f1_incr_thld_val('h0), .counter_f1_incr_sat_val('h0),
.block_size('h2000)) timer_ids_inst(
    .irq(),
    .status_reset(),
    .result_reset(),
    . . . . . .
    . . . . . .
    . . . . . .
    .pready(wire_timer_ids_inst_pready_block1_top_aggregation_APB_insttimer_ids_inst_ids_pready),
    .prdata(wire_timer_ids_inst_prdata_block1_top_aggregation_APB_insttimer_ids_inst_ids_prdata),
    .pslverr(wire_timer_ids_inst_pslverr_block1_top_aggregation_APB_insttimer_ids_inst_ids_pslverr)

);
endmodule
```

SystemVerilog output


```
include "../input/gpio.v"
include "ids_block1_top_aggregation_APB_inst_apb_aggregation.v"
include "../input/timer.v"

module block1_top #(
        parameter addr_width = 'h20,
        parameter bus_width = 'h32,
        parameter block1_top_offset = 'h0,
        parameter gpio_ids_address_width = 'hc,
        parameter gpio_ids_offset = 'h0,
        parameter timer_ids_address_width = 'hd,
        parameter timer_ids_offset = 'h0

    ) (. . . . . .
       . . . . . .
       . . . . . .)

);

ids_block1_top_aggregation_APB_inst_apb_aggregation #( .addr_width('h20),
.bus_width('h32), .block1_top_aggregation_APB_inst_offset(block1_top_offset),
.timer_ids_inst_addr_width(timer_ids_address_width),
.gpio_ids_inst_addr_width(gpio_ids_address_width)) block1_top_aggregation_APB_inst(
    .pclk(pclk),
    .presetn(presetn),
    . . . . . .
    . . . . . .
    .gpio_ids_inst_ids_prdata(wire_gpio_ids_inst_prdata_block1_top_aggregation_APB_instgpio_ids_inst_ids_prdata),
    .gpio_ids_inst_ids_pslverr(wire_gpio_ids_inst_pslverr_block1_top_aggregation_APB_instgpio_ids_inst_ids_pslverr)

);
timer_ids #( .NUM_TIMER('h1), .NUM_SRC('h8), .PRESCALER_WIDTH('h20), .COUNTER_WIDTH('h20),
.bus_width('h32), .addr_width(timer_ids_address_width), .block_offset(timer_ids_offset),
.counter_f1_incr_val('h0), .counter_f1_incr_thld_val('h0), .counter_f1_incr_sat_val('h0),
.block_size('h2000)) timer_ids_inst(
    .irq(),
    .status_reset(),
    .result_reset(),
    . . . . . .
    . . . . . .
    . . . . . .
    .pready(wire_timer_ids_inst_pready_block1_top_aggregation_APB_insttimer_ids_inst_ids_pready),
    .prdata(wire_timer_ids_inst_prdata_block1_top_aggregation_APB_insttimer_ids_inst_ids_prdata),
    .pslverr(wire_timer_ids_inst_pslverr_block1_top_aggregation_APB_insttimer_ids_inst_ids_pslverr)

);
endmodule
```

Benefits of using IP-XACT

- Standard allows multi vendor IPs/EDA tools use
- Simplified integration
- Coherency with other design teams
- Automatic flow to avoid manual repetitive jobs
- Dramatic Time to Market Improvements
- Documentation of all aspects of IP using an XML data book format
- Enables designers to deploy specialist knowledge in their design

References

- http://statustool.com/wiki/uploads/Site/IP-XACT%20standard%201685-2014.pdf
- https://www.accellera.org/images/downloads/standards/ip-xact/IP-XACT_User_Guide_2018-02-16.pdf
- https://www.researchgate.net/figure/IP-XACT-concepts-for-a-component-description_fig4_235345384
- https://www.researchgate.net/figure/IP-XACT-Design-Environment-and-supported-XML-schemas_fig16_265125404
- https://www.design-reuse.com/articles/18613/ip-xact-esl-noc.html
- https://www.slideshare.net/DVClub/verification-automation-using-ipxact

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Lab Exercises

Lab 1

- Create a Component in IP-XACT

```
<spirit:component xmlns:ids="http://www.agnisys.com/"
   xmlns:snps="http://www.synopsys.com/"
   xmlns:xrsl="http://www.agnisys.com/"
   xmlns:spirit="http://www.spiritconsortium.org/XMLSchema/SPIRIT/1685-2009">
   <spirit:vendor>Agnisys_Inc.</spirit:vendor>
   <spirit:library>Agnisys_Inc.</spirit:library>
   <spirit:name>block_name_model</spirit:name>
   <spirit:version>1.0</spirit:version>
   <spirit:busInterfaces> [101 lines]
   <spirit:memoryMaps> [54 lines]
   <spirit:model> [137 lines]
   <spirit:vendorExtensions> [2 lines]
</spirit:component>
```

VLNV

Next slide

```
<spirit:busInterfaces>
   <spirit:busInterface>
      <spirit:name>default_map</spirit:name>
      <spirit:busType spirit:vendor="AGNISYS"
         spirit:library="AGNISYS" spirit:name="CUSTOM"
         spirit:version="1.0"/>
      <spirit:abstractionType spirit:vendor="AGNISYS"
         spirit:library="AGNISYS"
         spirit:name="CUSTOM_rtl" spirit:version="1.0"/>
      <spirit:slave> [2 lines]
      <spirit:portMaps> [89 lines]
   </spirit:busInterface>
```

```
<spirit:portMaps>
   <spirit:portMap>
      <spirit:logicalPort>
         <spirit:name>clk</spirit:name>
      </spirit:logicalPort>
      <spirit:physicalPort>
         <spirit:name>clk</spirit:name>
      </spirit:physicalPort>
   </spirit:portMap>
</spirit:portMaps>
```

```
<spirit:slave>
   <spirit:memoryMapRef spirit:memoryMapRef="block_namemap"/>
</spirit:slave>
```

Lab 1

memoryMap

```
<spirit:memoryMaps>
   <spirit:memoryMap>
      <spirit:name>block_namemap</spirit:name>
      <spirit:addressBlock> [39 lines]
      <spirit:addressUnitBits>8</spirit:addressUnitBits>
      <spirit:vendorExtensions> [8 lines]
   </spirit:memoryMap>
</spirit:memoryMaps>
```

model

```
<spirit:model>
   <spirit:ports>
      <spirit:port>
         <spirit:name>clk</spirit:name>
         <spirit:wire>
            <spirit:direction>in</spirit:direction>
         </spirit:wire>
      </spirit:port>
      <spirit:port> [5 lines]
      <spirit:port>
         <spirit:name>block_name_idsreg_name_F1_r</spirit:name>
         <spirit:wire>
            <spirit:direction>out</spirit:direction>
            <spirit:vector>
               <spirit:left>31 </spirit:left>
               <spirit:right> 0</spirit:right>
            </spirit:vector>
         </spirit:wire>
      </spirit:port>
   </spirit:ports>
   <spirit:modelParameters> [11 lines]
</spirit:model>
```

```
<spirit:addressBlock>
   <spirit:name>block_ipxact</spirit:name>
   <spirit:description/>
   <!-- // [1 line]
   <spirit:baseAddress>0x0</spirit:baseAddress>
   <spirit:range>0x8</spirit:range>
   <spirit:width>32</spirit:width>
   <spirit:register>
      <spirit:name>reg_name</spirit:name>
      <spirit:addressOffset>0x4</spirit:addressOffset>
      <spirit:size>32</spirit:size>
      <spirit:volatile>true</spirit:volatile>
      <spirit:reset> [3 lines]
      <spirit:field> [9 lines]
      <spirit:vendorExtensions> [4 lines]
   </spirit:register>
   <spirit:vendorExtensions> [5 lines]
</spirit:addressBlock>
```

```
<spirit:field>
   <spirit:name>F1</spirit:name>
   <spirit:bitOffset>0</spirit:bitOffset>
   <spirit:bitWidth>32</spirit:bitWidth>
   <spirit:volatile>true</spirit:volatile>
   <spirit:access>read-write</spirit:access>
   <spirit:vendorExtensions>
      <ids:default_value>0</ids:default_value>
   </spirit:vendorExtensions>
</spirit:field>
```

```
<spirit:reset>
   <spirit:value>0x00000000</spirit:value>
   <spirit:mask>0xFFFFFFFF</spirit:mask>
</spirit:reset>
```

vendorExtensions

```
<spirit:vendorExtensions>
   <ids:type>false</ids:type>
</spirit:vendorExtensions>
```

Lab 2

- Use TGI to create a component register description in an address block of 4K 32-bit words

```
# Create a new component with the given VLNV and get its unconfigured ID
   set componentID [ tgi::createComponent [ list "accellera.org" "myLib" "myComponent" "1.0" ] ]

# Create a new memory map in the component with the given memory map name
   set memoryMapID [ tgi::addComponentMemoryMap $componentID "myMemoryMap" ]

# Create a new address block in the memory map with the given address block name, base address,
    range, and width
   set addressBlockID [ tgi::addMemoryMapAddressBlock $memoryMapID "myAddressBlock" "'h0" "'h1000" "32" ]

# Create a new register in the address block with the given register name, address offset, and size,
    and create a new field in the register with the given field name, bit offset, and bit width
   set registerID [ tgi::addAddressBlockRegister $addressBlockID "reg" "'h0" "32" "field1" "0" "8" ]

# Set the description, access, reset value, and reset mask of the register
   tgi::setDescription $registerID "This is my register description."
   tgi::setRegisterAccess $registerID "read-write"
   tgi::setRegisterResetValue $registerID "'h1" ""
   tgi::setRegisterResetMask $registerID "'hffffffff" ""

# Create a new field in the register with the given field name, bit offset, and bit width
   set fieldID [ tgi::addRegisterField $registerID "field2" "8" 24 ]

# Set the description, access, read action, and volatility of the field
   tgi::setDescription $fieldID "This is my second field description"
   tgi::setRegisterFieldAccess $fieldID "read-only"
   tgi::setRegisterFieldReadAction $fieldID "clear"
   tgi::setRegisterFieldVolatility $fieldID "true"
```

Lab 3

- Use TGI to traverse the component memory maps for accessing the registers in address blocks

```
# Get unconfigured ID for the component with the given VLNV
set componentID [ tgi::getID [ list "accellera.org" "myLib" "myComponent" "1.0" ] ]
 # Get the memory maps of the component
  set memoryMapIDs [ tgi::getComponentMemoryMapIDs $componentID ]
   # Walk each memory map
      foreach memoryMapID $memoryMapIDs {
         # Get the memory map elements of the memory map
           set memoryMapElementIDs [ tgi::getMemoryMapElementIDs $memoryMapID ]
              # Walk each memory map element
                 foreach memoryMapElementID $memoryMapElementIDs {
                      # Get the type of the memory map element
                         set type [ tgi::getMemoryMapElementType $memoryMapElementID ]
                      # Check if the memory map element is an address block
                         if { [ string compare $type "addressBlock" ] == 0 } {
                    # Get the registers of the address block
                set registerIDs [ tgi::getAddressBlockRegisterIDs $memoryMapElementID ]
                      # Walk each register
                         foreach registerID $registerIDs {
                            # Get the register name, description, address offset, access, reset value, and reset mask
                                 set registerName [ tgi::getName $registerID ]
                                 set registerDescription [ tgi::getDescription $registerID ]
                                 set registerOffset [ tgi::getRegisterAddressOffset $registerID ]
                                 set registerAccess [ tgi::getRegisterAccess $registerID ]
                                 set registerResetValue [ tgi::getRegisterResetValue $registerID ]
                                 set registerResetMask [ tgi::getRegisterResetMask $registerID ]
                             }
                         }
                      }
                   }
```

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

<thankYou></thankYou>