SCIpX User Manual

Draft version 0.0

Benoît Ferrero
EPI AOSTE
INRIA Sophia Méditerranée

This draft document provides user guidance for potential users of the SCIpX (pronounce [saï pix]) tool. The tool extractx IP-XACT models (IEEE 1685 standard) from SystemC files (IEEE 1666 standard). Currently the tool focuses on structural component descriptions and interconnects.

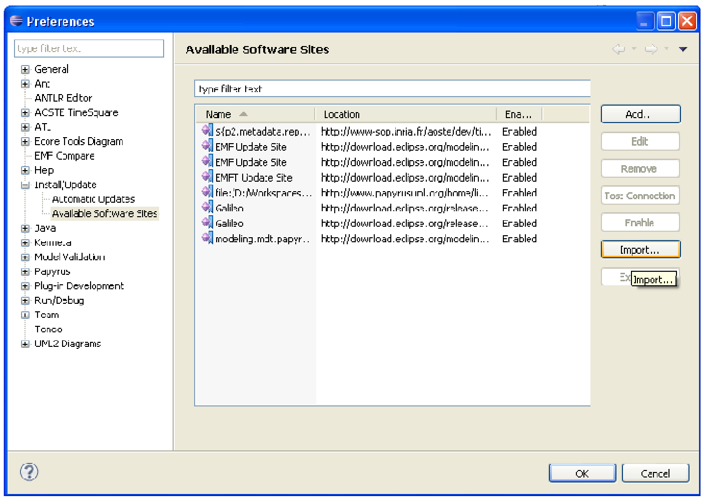
SCIpX is based both on PinaVM for symbolic elaboration, and Doxygen for static analysis, whose results are later combined and translated into IP-XACT files.

The document describes the various format assumptions and requirements for the input and output files. IP-XACT representations come as Ecore models. The main features of the various transformations are outlined, and the practical way to apply these successive transformation steps is also described.

1 Install

`SCIpX TimeSquare` may be installed over an existing Eclipse installation using the following update site:

- *Help >; Install New Software >*
- *Available Software Site ... >*
    - o Download [BookMark File (on http://www-sop.inria.fr/aoste/software/ipxact2marte/bookmarks.xml)](http://www-sop.inria.fr/aoste/software/ipxact2marte/bookmarks.xml) and Import this File (*Import ...*)



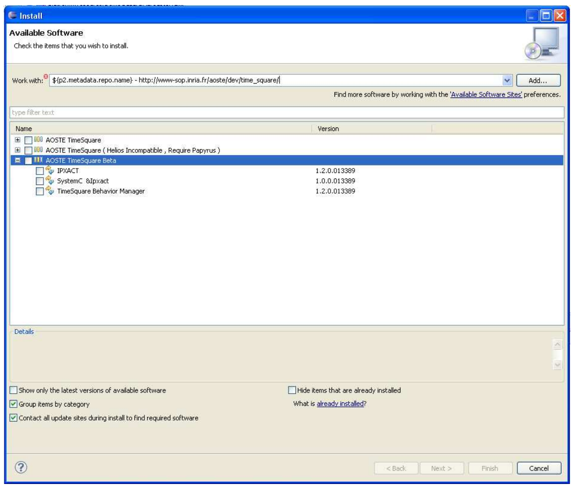
Available Software Sites

type filter text

Name	Location	Ena...
${p2.metadata.rep...	http://www-sop.inria.fr/aoste/dev/ti...	Enabled
EMF Update Site	http://download.eclipse.org/modelin...	Enabled
EMF Update Site	http://download.eclipse.org/modelin...	Enabled
EMFT Update Site	http://download.eclipse.org/modelin...	Enabled
file:/D:/Workspaces...	http://www.papyrusuml.org/home/li...	Enabled
Galileo	http://download.eclipse.org/release...	Enabled
Galileo	http://download.eclipse.org/release...	Enabled
modeling.mdt.papyr...	http://download.eclipse.org/modelin...	Enabled

- o Press Ok for return to "Install" Windows

- select site :_`http://www-sop.inria.fr/aoste/software/ipxact2marte/update/"`
- select desired features



Available Software

Check the items that you wish to install.

Work with: ${p2.metadata.repo.name} - http://www-sop.inria.fr/aoste/dev/time_square/

Find more software by working with the 'Available Software Sites' preferences.

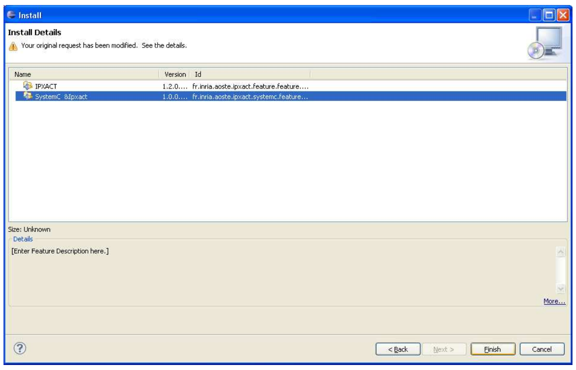
type filter text

Name	Version
AOSTE TimeSquare	
AOSTE TimeSquare (Helios Incompatible , Require Papyrus)	
AOSTE TimeSquare Beta	
IPXACT	1.2.0.013389
SystemC &Ipxact	1.0.0.013389
TimeSquare Behavior Manager	1.2.0.013389

Details

- [ ] Show only the latest versions of available software
- [✓] Group items by category
- [✓] Contact all update sites during install to find required software
- [ ] Hide items that are already installed

What is already installed?



Install Details

Your original request has been modified. See the details.

Name	Version	Id
IPXACT	1.2.0....	fr.inria.aoste.ipxact.feature.feature....
SystemC &Ipxact	1.0.0....	fr.inria.aoste.ipxact.systemc.feature...

Size: Unknown

Details

[Enter Feature Description here.]

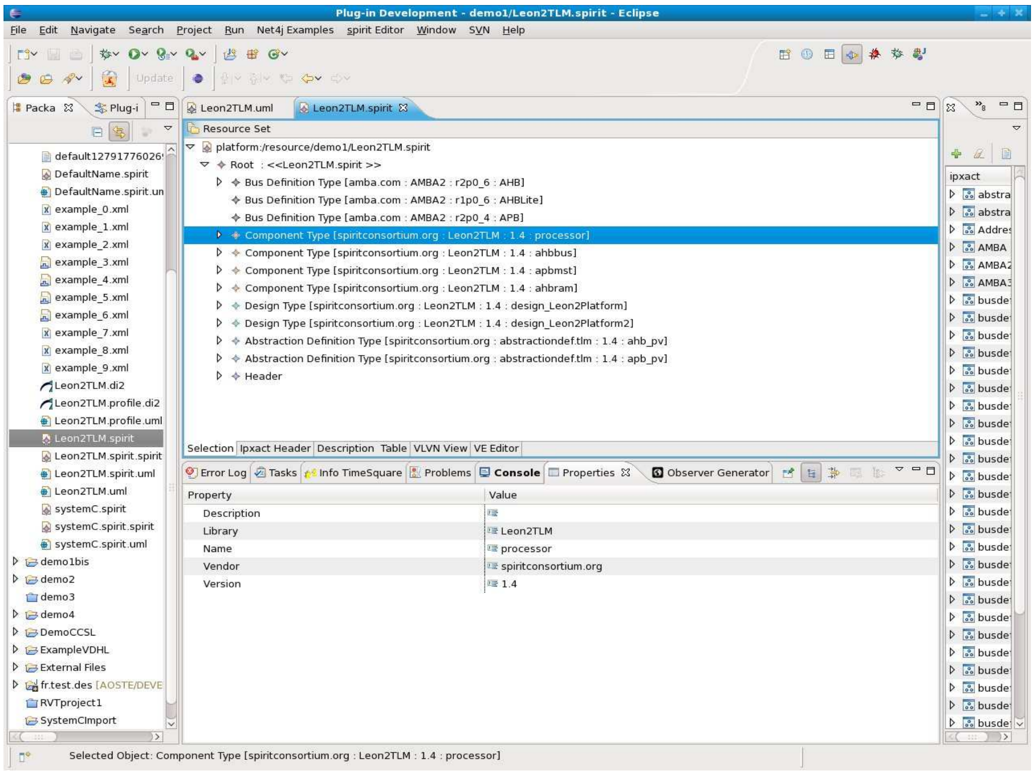
2 Ecore.IPXACT Model

1 Presentation

The ECORE.IPXACT model is Ecore model which   representing the concepts of IPXACT.

It allows multiple objects IPXACT (Component, Design, AbstractionDefinion and BusDefinition) in the same file.
The tool provides a mechanism to import and export IPXACT.
It includes a system for automatic link resolution (LibraryRef), Management Vendor Extensions.

2 Editor



```
platform:/resource/demo1/Leon2TLM.spirit
  Root  : <<Leon2TLM.spirit >>
    Bus Definition Type [amba.com : AMBA2 : r2p0_6 : AHB]
    Bus Definition Type [amba.com : AMBA2 : r1p0_6 : AHBLite]
    Bus Definition Type [amba.com : AMBA2 : r2p0_4 : APB]
    Component Type [spiritconsortium.org : Leon2TLM : 1.4 : processor]
    Component Type [spiritconsortium.org : Leon2TLM : 1.4 : ahbbus]
    Component Type [spiritconsortium.org : Leon2TLM : 1.4 : apbmst]
    Component Type [spiritconsortium.org : Leon2TLM : 1.4 : ahbram]
    Design Type [spiritconsortium.org : Leon2TLM : 1.4 : design_Leon2Platform]
    Design Type [spiritconsortium.org : Leon2TLM : 1.4 : design_Leon2Platform2]
    Abstraction Definition Type [spiritconsortium.org : abstractiondef.tlm : 1.4 : ahb_pv]
    Abstraction Definition Type [spiritconsortium.org : abstractiondef.tlm : 1.4 : apb_pv]
    Header
```

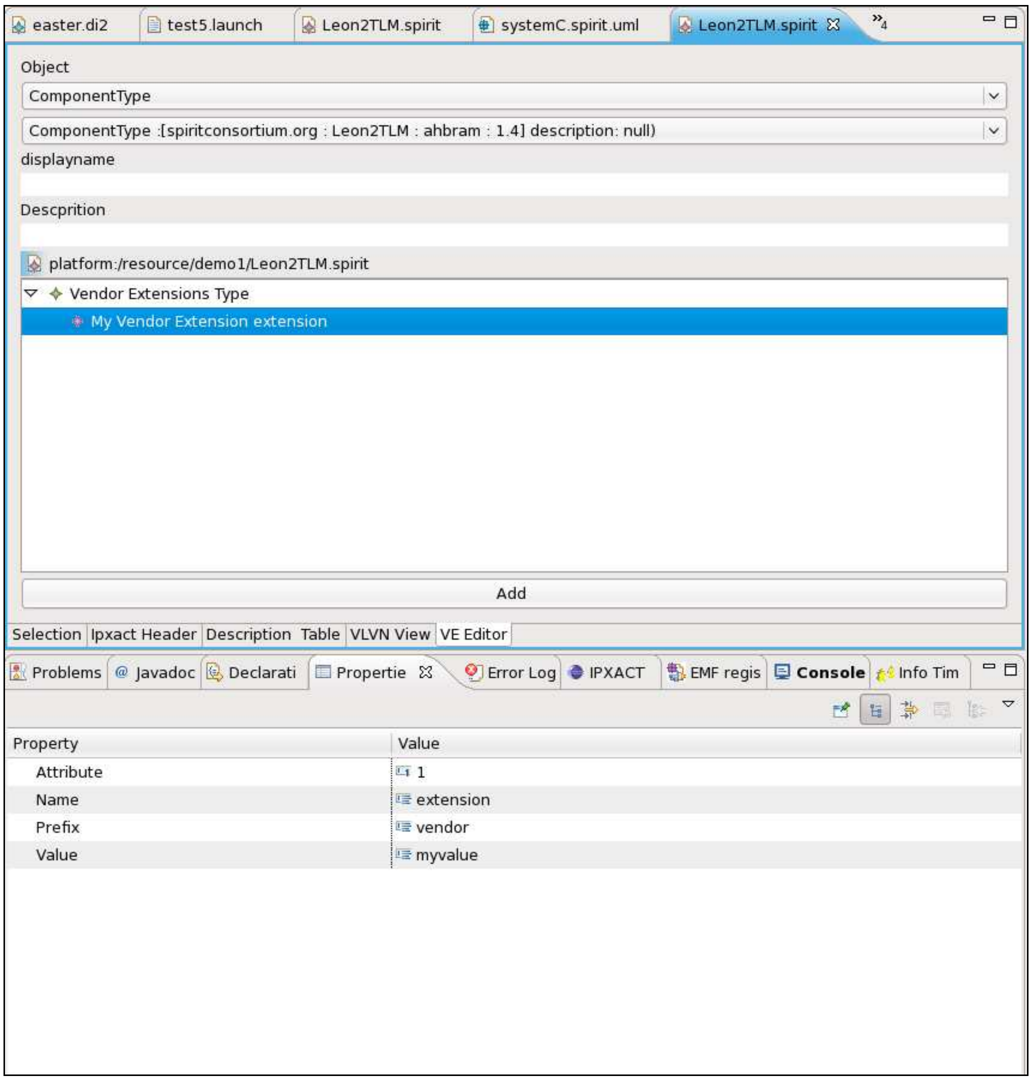
Property	Value
Description	
Library	Leon2TLM
Name	processor
Vendor	spiritconsortium.org
Version	1.4

Selected Object: Component Type [spiritconsortium.org : Leon2TLM : 1.4 : processor]

- Multi-View Editor
- Import /Export IPXACT
- Resolution Automatic links (LibraryRef : update of VLNV Fields when the target changed)
- Editor with view for vendor-extensions.

Vendor Extensions View

The editor view is dedicated to the Vendor-extension allows to edit.



easter.di2 | test5.launch | Leon2TLM.spirit | systemC.spirit.uml | Leon2TLM.spirit

Object

ComponentType

ComponentType :[spiritconsortium.org : Leon2TLM : ahbram : 1.4] description: null)

displayname

Descprition

platform:/resource/demo1/Leon2TLM.spirit

Vendor Extensions Type

My Vendor Extension extension

Add

Selection | Ipxact Header | Description | Table | VLVN View | VE Editor

Problems | Javadoc | Declarati | Propertie | Error Log | IPXACT | EMF regis | Console | Info Tim

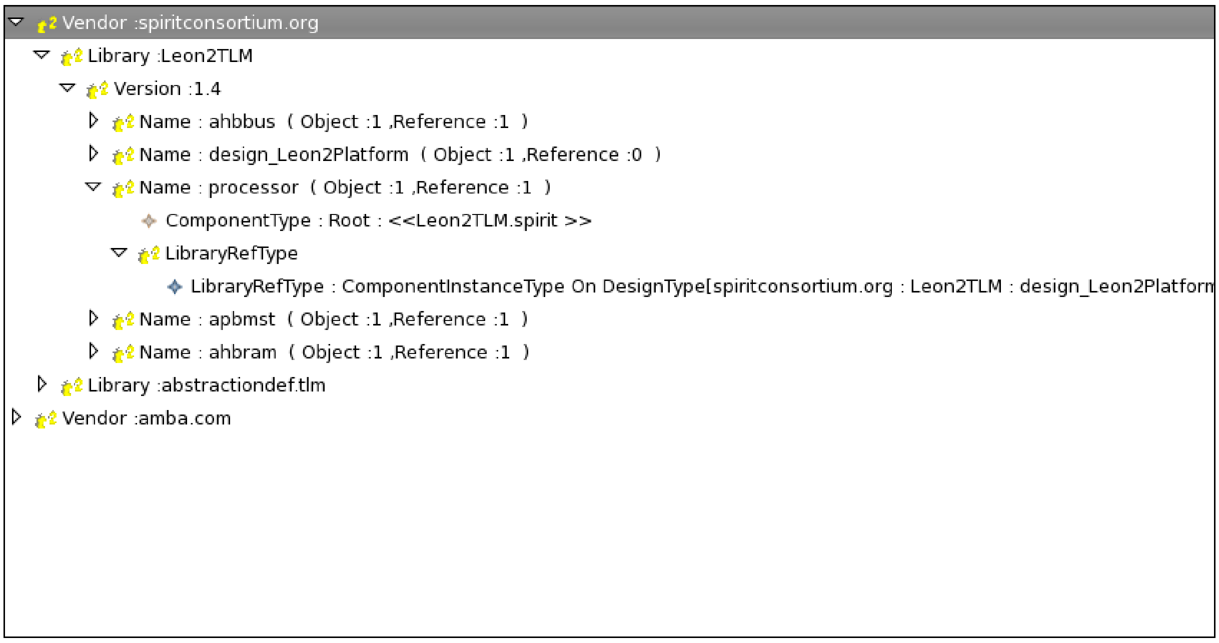
Property	Value
Attribute	1
Name	extension
Prefix	vendor
Value	myvalue

1Vendor Editor View

For each vendor extension, one can define its name (prefix and name), its value, and type (Attribute, *in* the sense of XML, 1: **attribute**, 2: **node**, 3: **text**).

VLNV View

In VLNV view, display object in the tree. The tool also lists all LibraryRef for a given target object.



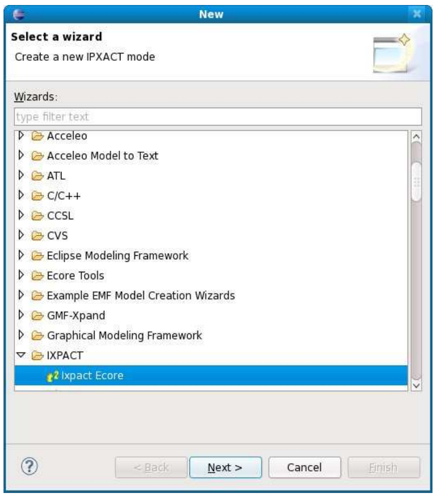
```
▽ Vendor :spiritconsortium.org
  ▽ Library :Leon2TLM
    ▽ Version :1.4
      ▷ Name : ahbbus  ( Object :1 ,Reference :1  )
      ▷ Name : design_Leon2Platform  ( Object :1 ,Reference :0  )
      ▽ Name : processor  ( Object :1 ,Reference :1  )
          ComponentType : Root : <<Leon2TLM.spirit >>
        ▽ LibraryRefType
            LibraryRefType : ComponentInstanceType On DesignType[spiritconsortium.org : Leon2TLM : design_Leon2Platforn
      ▷ Name : apbmst  ( Object :1 ,Reference :1  )
      ▷ Name : ahbram  ( Object :1 ,Reference :1  )
  ▷ Library :abstractiondef.tlm
▷ Vendor :amba.com
```

2 VLVN view of editor
Example: here you have a processor (component [spiritconsortium.org:Leon2TLM:processor:1.4]) whose use in design : ([spiritconsortium.org:Leon2TLM:design_Leon2Platform:1.4])

3 Create Ecore.IPXACT Model

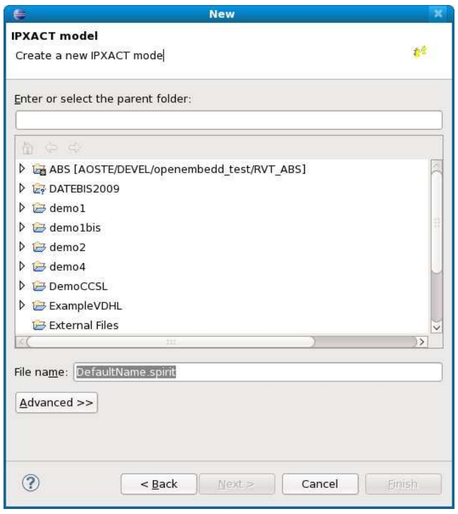
For create a new model, launch wizard (click to **File** >> **New** >>**Other...** **(Ctrl +N))**

First page of the wizard



Select IPXACT / IPXACT Ecore and Click to *Next >*

Second page of the wizard



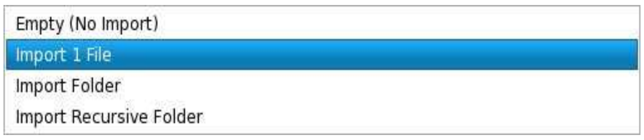
File name: DefaultName.spirit

Select the parent folder and given the file name: (cf. eclipse help: Workbench User Guide: File)

Third page of the wizard

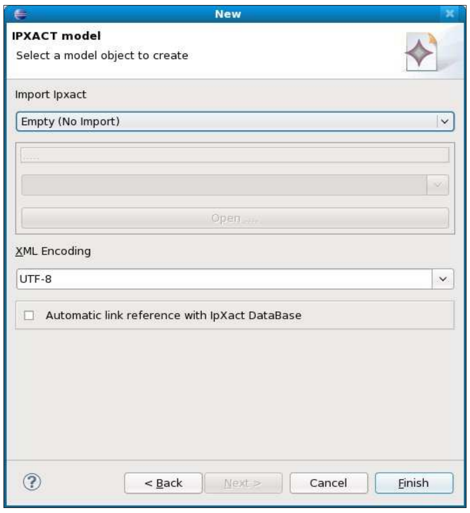
The third page of wizard can choose mode of model creation.

There are 4 modes.



Empty (No Import)
Import 1 File
Import Folder
Import Recursive Folder

Depending on the choice of mode, the fields in the dialog evolve.



Click to Finish, the document is created. The editor opens automatically.

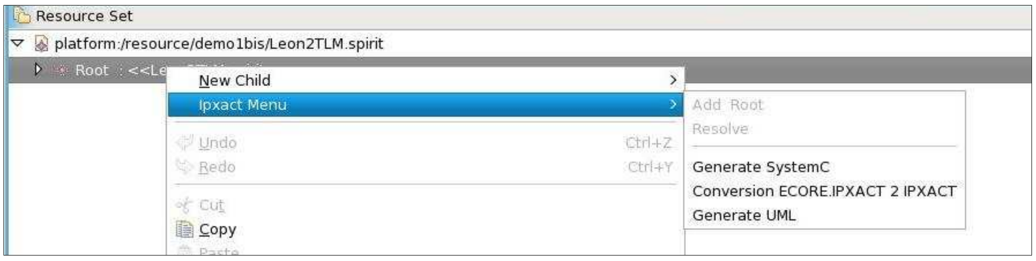
4 Export to IPXACT

Export Ecore.Ipxact model and convert to IPXACT file.

Running export

- **In Ecore.IPXACT editor**

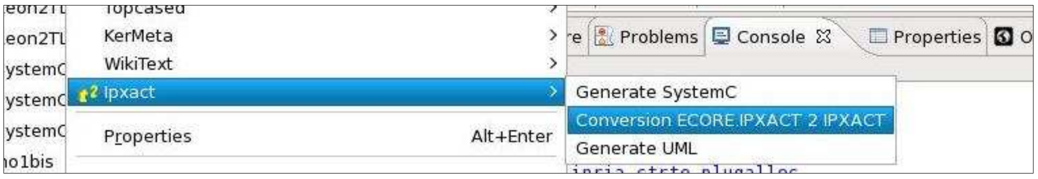
Select an element in model, and in its context menu, use a command: **IPXACT Menu >> Conversion Ecore. IPXACT 2 IPXACT**



Caution: you export the current document with the latest changes even if they have not saved.

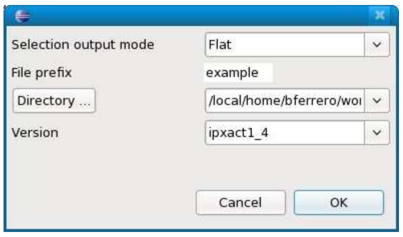
- **In the Workspace**

In the Workspace, select file * .spirit, and in its context menu, use a command: IPXACT >> Conversion Ecore. IPXACT 2 IPXACT



Export option

When you launch export, a dialog box show up.



Selection output mode: Flat
File prefix: example
Directory ...: /local/home/bferrero/wo
Version: ipxact1_4

The dialogue box will request four information:

- **The Output mode : (***Flat* **or** *Hierarchic***)**
- **The file name prefix (when the Output mode is : *Flat*)**
- **The output directory where the files will be written**
- **Version of IPXACT (1.4 or 1.5)**

There are 2 modes of output:

- **Flat:**

  In the output directory, it creates file which name starting with prefix and follow by a number.

  example : in the model , you have a Design[*"myvendor"* , *"mylibray"* , *"mydesign"* , *"version"*] and a Component [*"componantvendor"* ,*"componantlibrary"* ,*"componantname"* , *"componantversion"*]

  if you choose, a output directory : *myfolder* and prefix : *myprefix*

  it creates 2 files :
  For a Component : my*folder/myprefix_0.xml*
  For a Design : *myfolder/myprefix_1.xml*

- **Hierarchic : in the folder : *outputFolder* :**

  For each «object" VLNV [vendor, library, name, version], it creates a folder:
  %outputFolder%/%vendor%/%library%/%name%/%version%
  in this folder, it creates a file %name%.xml

  For a same example and the output directory is *myfolder*

  it creates 2 files :
  For a Design : *myfolder/myvendor/mylibray/mydesign/version/mydesign.xml*
  For a Component :
  myfolder/componantvendor/componantlibrary/componantname/componantversion/componantname.xml

3 Transformation FromSystemC Source code to Ecore.IPXACT

Using two intermediate models: SystemC Model and Doxygene.

1 Create Ecore.IPXACT Model FromSystemC source code

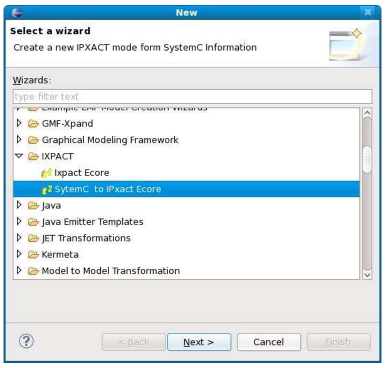
To create a new document from a SystemC source code

- Running tool (based to pinavm and doxygen, en dehors d'Eclipse) (link ??)

Back to Eclipse:

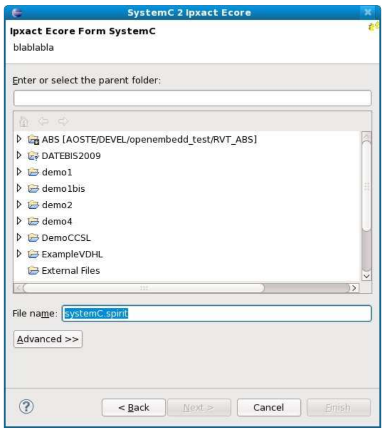
Click to File >> New >> Other... (Ctrl +N)

First page of the wizard



Select **IPXACT /** SystemC to IPXACT Ecore and Click to *Next >*

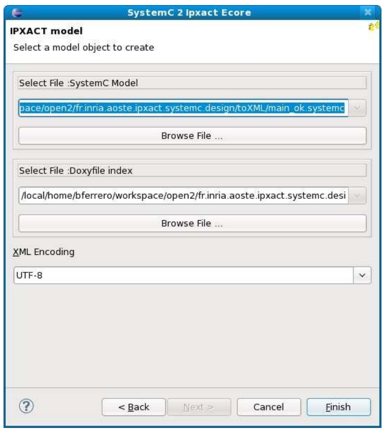
Second page of the wizard



File name: systemC.spirit

Select the parent folder and given the file name: (cf. eclipse help: Workbench User Guide: File Wizard)

Third page of the wizard


SystemC 2 Ipxact Ecore

IPXACT model

Select a model object to create

Select File :SystemC Model

pace/open2/fr.inria.aoste.ipxact.systemc.design/toXML/main_ok.systemc

Select File :Doxyfile index

/local/home/bferrero/workspace/open2/fr.inria.aoste.ipxact.systemc.desi

XML Encoding

UTF-8

Select 2 files:

(Created by the tool based on pinavm and Doxygen)

- **SystemC Model**: file *.*systemc* (Dynamics Aspect ==> for create a Design).
- **Doxyfile Index**: file *index.coumpound* (Static Aspect ==> for create a Component).

2 Rule Creation

SystemCModel		
	SystemC	IPXACT!Design
	• name (if null : "noname")	VLNV {"vendor" ,"lib" , %name% , "version" }
	Modules M (Type t<P>) • name • typeName : t.name if the module have array of port , we extract the size of each array to make a parameter P	ComponentInstance M (de Component T) • instance Name • componentRef --> Component
	Channel List of port	Connection ad-hoc (name) Port InternalPort (portRef, componentRef)

Doxygene		
	Type (Name) Combine with Parameter (if present)	IPXACT!Component
		VLNV ["vendor" ,"lib" , %name% , "version"] %name%= T.name + Parameter
	List of the member (**public-attrib**) • name • definition • definition (extraction du type) • if extend *sc_in* • if extend *sc_out* • if extend *sc_inout* (but not *sc_out*) • if there is a parameter that gives the array size N	Port • name • description • Port Wire : • direction <= In • direction <= Out • direction <= InOut • vector <= [0 : N-1]

4 Transformation UML (with «IPXACT4UML" Profile) -> IPXACT

1 Using a profile

Apply a profile

For apply a profile.

In UML Model Editor:

1. Load a Profile ("UML Editor " >> "Load Resource…») :"***pathmap://SPIRITUML/IPXACT4Uml.profile.uml***"
2. for each package concerned,
    1. Select this
    2. Apply a profile IPXACT4UML ("UML Editor» >> "Package» >> "Apply Profile").

or

In Papyrus UML2 Editor:

1. Select a package.
2. Go to "Profile" tab of "Properties" view,
3. Load and apply a Profile ("*Apply registered profile...*") "IPXACT4UML",

Using a Profile

Creating a BusDefinition:

1. Create a UML!Package which name is a library name
2. Apply a profileIPXACT4UML
3. Create a UML!Class
4. Apply a stereotype *<<BusDefinition>>*
5. if the BusDefinition extends other BusDefinition, set the field *<<BusDefinition>>.extends* with this BusDefinition (Optional)
6. set the field *<<BusDefinition>>.ident* with a String of type: "(vendor ='%vendor%' , version='%version%')" where %vendor%, and %version% are identifier of *vendor* and *version* .
7. set the maximum number of master on the bus in *<<BusDefinition>>.maxMasters* (Optional)
8. set the maximum number of slave on the bus in *<<BusDefinition>>.maxSlaves* (Optional)
9. given a description *<<BusDefinition>>.description* (Optional)

Creating an AbstractionDefinition:

1. Create a UML! Package which name is a library name
2. Apply a profileIPXACT4UML
3. Create a UML!Class
4. Apply a stereotype *<<AbstractionDefinition>>*
5. If the AbtractionDefinition extends other AbtractionDefinition, set the field *<<AbstractionDefinition>>.extends* with this AbtractionDefinition (Optional)
6. Set the field *<<AbstractionDefinition>>.ident* with a String of type: "(vendor ='%vendor%' , version='%version%')" where %vendor%, and %version% are identifier of *vendor* and *version*.
7. Set the field *<<AbstractionDefinition>>.bustype* with a BusDefinition
8. Give a description *<<AbstractionDefinition>>.description* (Optional)
9. Add an AbstractionPort Transactional *or* an AbstractionPort Wire.

- **Creating an AbstractionPort Transactional**

1. From an AbstractionDefinition, create a UML!Class
2. Apply a stereotype *<<AbsTransactionalPort>>*
3. Set the field *<<AbsTransactionalPort>>.isAddress*
4. Set the field *<<AbsTransactionalPort>>.isData*
5. Set the field *<<AbsTransactionalPort>>.PresenceOnMaster* ((Enumeration Presence: *none, required, illegal, optional*)
6. Set the field *<<AbsTransactionalPort>>.ServicetypeOnMaster* (name of Class<<PortTransactionalType>>)
   There is a wizard to configure the information.
7. Set the field *<<AbsTransactionalPort>>.ServicetypeImplicitOnMaster* (Boolean)
8. Set the field *<<AbsTransactionalPort>>.InitiativeOnMaster* (Enumeration Initiative : *none, requires, provides, both, phatom*)
9. Set the field *<<AbsTransactionalPort>>.PresenceOnSlave* (Enumeration Presence: *none, required, illegal, optional*)
10. Set the field *<<AbsTransactionalPort>>.ServicetypeOnSlave* (name of children Class of PortTransactionalType)
11. Set the field *<<AbsTransactionalPort>>.ServicetypeImplicitOnSlave* (Boolean)
12. Set the field *<<AbsTransactionalPort>>.InitiativeOnSlave* (Enumeration Initiative: *none, requires, provides, both, phatom*)

- **Creating an AbstractionPort Wire**

1. From an AbstractionDefinition, create a UML!Class
2. Apply a stereotype *<<AbsWirePort>>*
3. Set the field *<<AbsWirePort>>.isAddress*
4. Set the field *<<AbsWirePort>>.isData*
5. Set the field *<<AbsWirePort>>.isClock*
6. Set the field *<<AbsWirePort>>.isReset*
7. Set the field *<<AbsWirePort>>.PresenceOnMaster* (Enumeration Presence: *none, required, illegal, optional*)
8. Set the field *<<AbsWirePort>>.PresenceOnSlave* (Enumeration Presence: *none, required, illegal, optional*)

Creating a Component:

1. Create a UML!Package which name is a library name
2. Apply a profileIPXACT4UML
3. In this Package, create a UML!Component
4. Give it a name
5. You may adding a PortTransactionalType *ou* PortWireType
6. You may adding a Businterfaces

- **Creating a PortTransactionalType**

1. From a Component, create a UML!Class
2. Give it a name
3. Apply a stereotype *<<PortTransactionalType>>*

- **Creating a PortWireType**

1. From a Component , create a UML!Class
2. Give it a name
3. Apply a stereotype *<<PortWireType>>*
4. Set direction in *<<PortWireType>>.direction* (*none, in, out, inout, phatom*)

Creating a BusInterface

1. From a Component , create a UML!Port

2. Give it a name.
3. Apply a stereotype *<<BusInterface>>*.
4. Set the field *<<BusInterface>>.bustype* with a Class<<BusDefinition>>.

Set the field *<<BusInterface>>.abstractiontype* with a Class <<AbstractionDefinition>>.

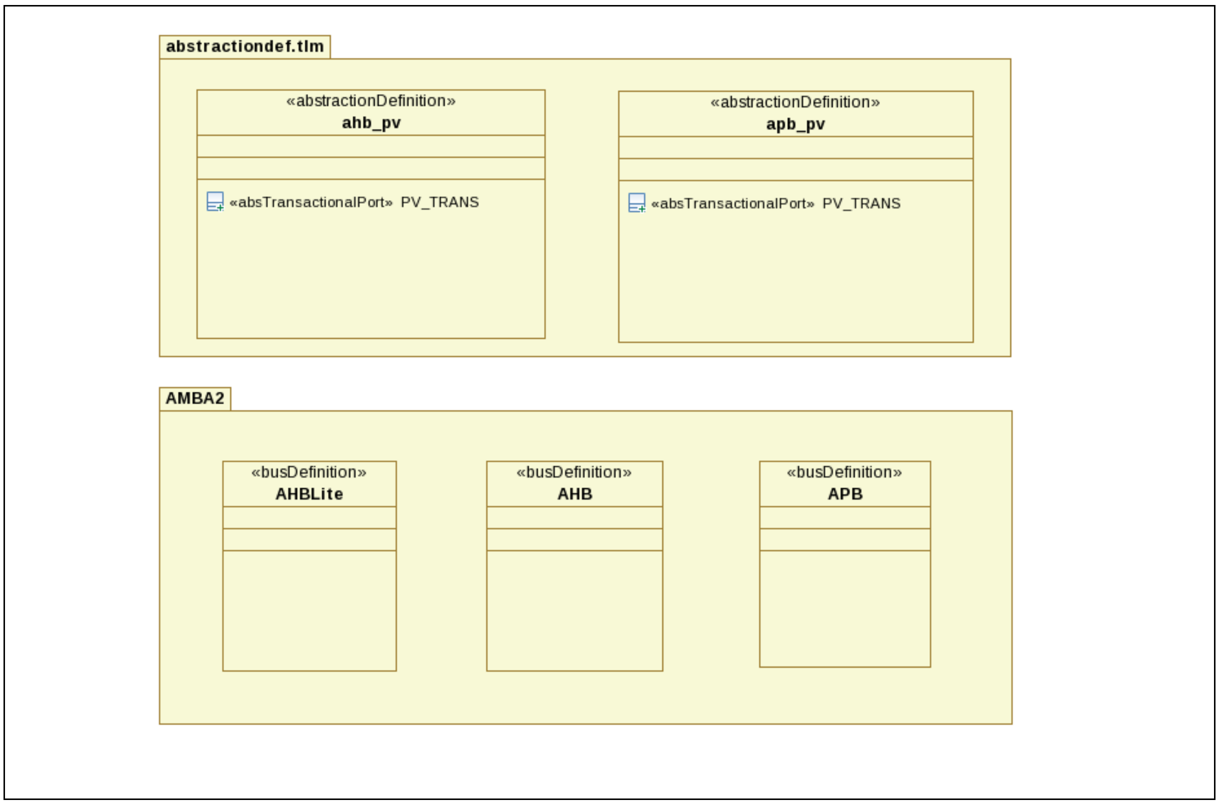
5. Set the enumeration *<<BusInterface>>.interfaceMode* (Enumeration *master,slave,system,mirroredMaster,mirroredSlave, mirroredSystem*)
6. give a list of PortMaps *<<BusInterface>>.portMaps*, for each PortMap , give a String :"(logicalPort='%logical%', physicalPort='%physical%')" where
    1. %logical% is name of PortTransactionalType and
    2. %physical% is name of PortWireType
       There is a wizard to configure the information.
7. Give a description *<<BusInterface>>.description* (Optional).
8. Give display Name *<<BusInterface>>.displayName* (Optional).

Creating a Design:

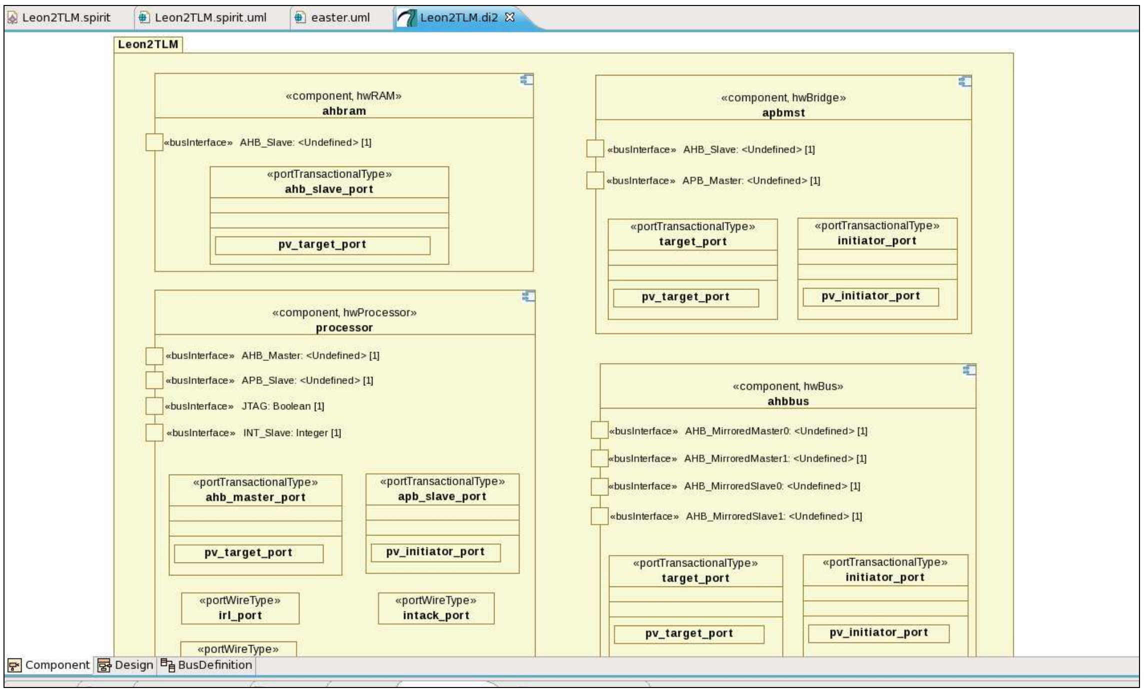
1. Create a UML!Package which name is a library name
2. In this package, create a UML!Class
3. Give it a name which starting by "design"
4. for each instance of component
    - create a UML!Proprety
    - give it a name
    - set type with a Component
5. for each link
    - Create a UML!Connector
    - Add UML!ConnectorEnd for each end point of a connector
        - Set ***PartWithPort*** (instance of the component owned a current design)
        - Set le ***Role*** (a port owned a component)

2 Example && Screenshots

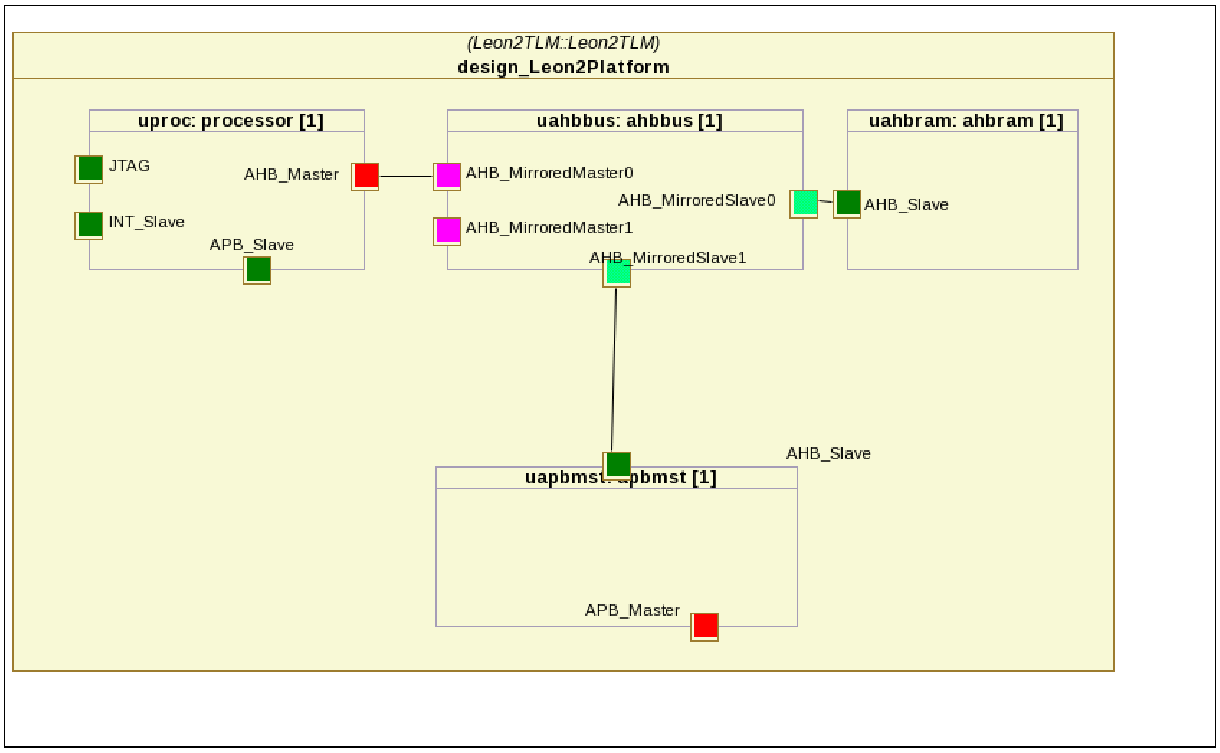
Here's an example based on LEON2. It is done under *Papyrus* and use a profile



3 BusDefinition


Leon2TLM.spirit
Leon2TLM.spirit.uml
easter.uml
Leon2TLM.di2
Leon2TLM
«component, hwRAM» ahbram
«busInterface» AHB_Slave: <Undefined> [1]
«portTransactionalType» ahb_slave_port
pv_target_port
«component, hwBridge» apbmst
«busInterface» AHB_Slave: <Undefined> [1]
«busInterface» APB_Master: <Undefined> [1]
«portTransactionalType» target_port
pv_target_port
«portTransactionalType» initiator_port
pv_initiator_port
«component, hwProcessor» processor
«busInterface» AHB_Master: <Undefined> [1]
«busInterface» APB_Slave: <Undefined> [1]
«busInterface» JTAG: Boolean [1]
«busInterface» INT_Slave: Integer [1]
«portTransactionalType» ahb_master_port
pv_target_port
«portTransactionalType» apb_slave_port
pv_initiator_port
«portWireType» irl_port
«portWireType» intack_port
«portWireType»
«component, hwBus» ahbbus
«busInterface» AHB_MirroredMaster0: <Undefined> [1]
«busInterface» AHB_MirroredMaster1: <Undefined> [1]
«busInterface» AHB_MirroredSlave0: <Undefined> [1]
«busInterface» AHB_MirroredSlave1: <Undefined> [1]
«portTransactionalType» target_port
pv_target_port
«portTransactionalType» initiator_port
pv_initiator_port
Component
Design
BusDefinition

4 Composant



5 Design

3 Transformation Rule:

	UML	IPXACT
***Design* :**	UML!Class *design* *the name must start with " **design"***	IPXACT!DesignType
		VLNV ["spiritconsortium.org", *design*.packageowner.name, *design*.name,"1.4"]
	• parts : {UML!Property}	• componentInstances :{ IPXACT!ComponentInstanceType}
	• ownedConnectors {UML!Connector}	• interconnections : {IPXACT!InterconnectionType}
	UML!Property *property*	IPXACT!ComponentInstanceType
	• name	• name <= property.name
	• type : Component	• componentRef <= IPXACT!LibraryRefType [VLNV (*property*.type)]
	UML!Connector *connector*	IPXACT!InterconnectionType
	• name	• name
	• ends • *ends[0]* • partWithRole (property du design) • role • *ends[1]* • partWithRole (property du design) • role	• activeinterface • IPXACT!Interface • componentRef<-*connector.ends*[0].partWithPort.name, • busRef<-*connector.ends[0]*.role.name • IPXACT!Interface • componentRef<-*connector.ends[1]*.partWithPort.name, • busRef<-*connector.ends[1]*.role.name

Component :	UML2!Component *component*	IPXACT!ComponentType
		VLNV['spiritconsortium.org',*component*.owner.name ,*component*.name ,'1.4']
	• ownedPorts : {UML2!Port}	• busInterfaces {IPXACT!BusInterfaceType}
	UML2!Port *port* must stereotyped by<<BusInterface>>	IPXACT!BusInterfaceType
	• *port*.name	• name
	• BusInterface.displayName (String ,*maybe null*)	• displayname
	• BusInterface.description (String ,*maybe null*)	• description
	• BusInterface.connectionRequired (Boolean)	• connectionRequired

	• BusInterface. bustype (<<BusDefintion>>)	• busType <= IPXACT!LibraryRefType [*VLNV (BusInterface.busType)*]
	• BusInterface.abstractionType (<<AbstractionDefinition>>)	• abstractionType <= IPXACT!LibraryRefType [*VLNV (BusInterface.abstractionType)*]
	• **BusInterface.interfaceMode (enumeration : master,slave,system, mirroredMaster,mirroredSlave, mirroredSystem)**	
	pour BusInterface.interfaceMode==master • *PortMap Rule*	• master <=IPXACT!MasterType()
	pour BusInterface.interfaceMode==slave	• slave <=IPXACT!SlaveType()
	pour BusInterface.interfaceMode==system • *PortMap Rule*	• system<=IPXACT!SystemType(group<=port.name)
	pour BusInterface.interfaceMode==mirroredMaster	• mirroredMaster <= MirroredMasterType()
	pour BusInterface.interfaceMode==mirroredSlave	• mirroredSlave<=IPXACT!MirroredSlaveType()
	pour BusInterface.interfaceMode==mirroredSystem • *PortMap Rule*	• mirroredSystem <= IPXACT!MirroredSystemType(group<=s.name)
	PortMap Rule: • BusInterface.portMaps: list of String. format strings are :"(logicalPort='%logicalname%' , physicalPort='%physicalname%')" with %logicalname% :name of Class <<AbsTransactionalPort>> %physicalname% :name of Class<<AbsWirePort>>	• portMaps : list of IPXACT!PortMapType for each String : there is: IPXACT!PortMapType(logicalPort<=IPXACT!LogicalPortType(name <= %logicalname%) ,physicalPort<= IPXACT!PhysicalPortType(name <= %physicalname%))

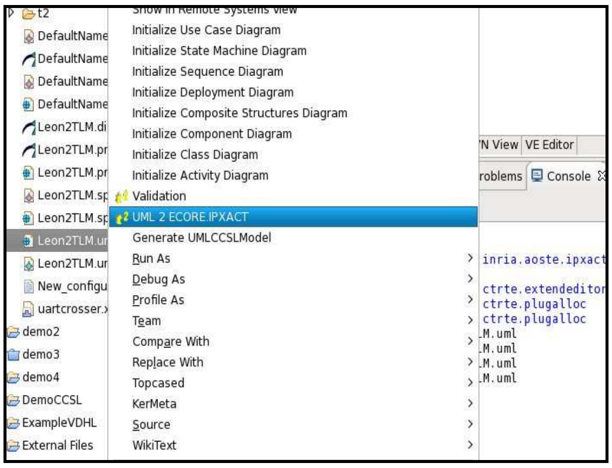
AbstractionDefinition:	UML!Class *class* Must stereotyped by <<AbstractionDefinition>>	IPXACT!AbstractionDefinition
	• AbstractionDefinition.ident (String : format :"vendor='%vendor%' , version='%version%' ")	VLNV[%vendor%, *class*.owner.name ,*class*.name, %version%]
	• AbstractionDefinition.busType (<<BusDefintion>>)	• bustype <= *IPXACT!LibraryRefType [VLNV (BusInterface.busType)]*
	• AbstractionDefinition.description (String, *maybe null*)	• description
	• AbstractionDefinition.extends (<<AbstractionDefinition>>, *maybe null*)	• extends <= *IPXACT!LibraryRefType [VLNV (BusInterface.extends)]*
	• nestedClassifer : {Classifier}	• ports <= {IPXACT!PortType1}
	UML!Classifier <<AbsTransactionalPort>> (cas transactionnel)	IPXACT!PortType1
	• name	• logicalName
	• AbsTransactionalPort.isAdress (Boolean) • AbsTransactionalPort.isData (Boolean) • AbsTransactionalPort.PresenceOnMaster (Enumeration : PresenceKind) • AbsTransactionalPort.InitiativeOnMaster (Enumeration : InitiativeKind) • AbsTransactionalPort.ServicetypeOnMaster (String) • AbsTransactionalPort.ServicetypeImplicitOnMaster (Boolean) • AbsTransactionalPort.PresenceOnSlave (Enumeration : PresenceKind) • AbsTransactionalPort.InitiativeOnSlave (Enumeration : InitiativeKind) • AbsTransactionalPort.ServicetypeOnSlave (String) • AbsTransactionalPort.ServicetypeImplicitOnSlave (Boolean)	• transactional <= IPXACT!TransactionalType(• qualifier<- IPXACT!QualifierType • isAddress <= AbsTransactionalPort.isAddress • isData <= AbsTransactionalPort'.isData • ,onMaster<=IPXACT!OnMasterType (• presence<= AbsTransactionalPort.PresenceOnMaster, • service = *_servicemaster)* • ,onSlave<=IPXACT!OnSlaveType (• presence<=AbsTransactionalPort'PresenceOnSlave • service <= *_serviceSlave))* Avec • _servicemaster<-IPXACT!ServiceType1 (• initiative<=AbsTransactionalPort.InitiativeOnMaster • typeName<= IPXACT!TypeNameType3 (• value<=AbsTransactionalPort.ServicetypeOnMaster .first() • ,implicit<=AbsTransactionalPort'.ServicetypeImplicitOnMaster .first()))) • _serviceslave<-IPXACT!ServiceType1 (• initiative<-AbsTransactionalPort.InitiativeOnSlave' • ,typeName<-IPXACT!TypeNameType3 (• value<- AbsTransactionalPort.ServicetypeOnSlave.first() • implicit<- AbsTransactionalPort.ServicetypeImplicitOnSlave .first()))

	UML!Classifier <<AbsWirePort>> (cas Wire)	IPXACT!PortType1
	• name	• logicalName
	• AbsWirePort.isAddress (Boolean) • AbsWirePort.isData (Boolean) • AbsWirePort.isClock (Boolean) • AbsWirePort.isReset (Boolean) • AbsWirePort.PresenceOnMaster (Enumeration :PresenceKind) • AbsWirePort.PresenceOnSlave (Enumeration: PresenceKind)	• wire<=IPXACT!WireType(• qualifier<- IPXACT!QualifierType1 • isAddress <= AbsWirePort.isAddress • isData <= AbsWirePort.isData • isClock <=AbsWirePort.isClock • isReset <= AbsWirePort.isReset) • onMaster <=IPXACT!OnMasterType1(• presence<=AbsWirePort.PresenceOnMaster) • onSlave <=IPXACT!OnSlaveType1(• presence<=AbsWirePort.PresenceOnSlave)
	UML!Enumeration InitiativeKind (none ,requires ,provides,both,phatom)	IPXACT!InitiativeType (requires ,provides,both,phatom)
	-none -requires -provides -both -phantom	-phantom -requires -provides -both -phantom
	UML!Enumeration PresenceKind (none,required,illegal,optional)	IPXACT!PresenceType (required,illegal,optional)
	-none -required -illegal -optional	-optional -required -illegal -optional

***BusDefinition*:**	UML!Class *class* must stereotyped by<< BusDefinition>>	IPXACT!BusDefinition
	• BusDefinition.ident (String : format is:"vendor='%vendor%' , version='%version%' ")	VLNV [%vendor%, *class*.owner.name, *class*.name, %version%]
	• BusDefinition.isAdressable (Boolean)	• isAdressable
	• BusDefinition.maxSlave (Integer ,*maybe null*)	• maxSlaves
	• BusDefinition.maxMaster (Integer ,*maybe null)*	• maxMaster
	• BusDefinition.directConnection (Boolean)	• directConnection
	• BusInterface.description (String, *maybe null*)	• description
	• BusDefinition.extends (<<BusDefintion>> ,*maybe null*)	• extends <= IPXACT!LibraryRefType [VLNV (BusInterface.extends)]

4 Launch Transformation:

In the Workspace, select file *.uml, and in its context menu, use a command UML 2 ECORE.IPXACT


Show in Remote Systems View
Initialize Use Case Diagram
Initialize State Machine Diagram
Initialize Sequence Diagram
Initialize Deployment Diagram
Initialize Composite Structures Diagram
Initialize Component Diagram
Initialize Class Diagram
Initialize Activity Diagram
Validation
UML 2 ECORE.IPXACT
Generate UMLCCSLModel
Run As
Debug As
Profile As
Team
Compare With
Replace With
Topcased
KerMeta
Source
WikiText

For a file *myfichier.uml*, you get a file *myfichier.spirit*. At the end of processing, the new file is automatically opened.

5 Transformation IPXACT -> UML

1 Transformation Rule

	IPXACT	UML avec Profile
Design :	**IPXACT!Design**	**UML!Class** ***design***
		Package:Class.owner.owner .name"design"
	VLNV [Library]	ownerPackage.name
	VLNV [Name]	name
	ComponentInstance *instance* • *instance*.name • *instance*.conponentRef	Property *property* • property.name = name • property.type <= Ref Component [VLNV]
	InterConnection : *interconnection* • *interconnection*.name • *interconnection*.activeInterface { } • componentRef • busRef	Connector *connector* • connector.name • connector.ends { } • partWithPort <= ref property [*instance.name* <= componentRef] • role <= ref port [Businterface.name <= busRef]

Component :	**IPXACT!Component [VLNV]**	**UML!Component**
		Package: Class.owner.owner.name= “component”
	VLNV [libray]	Package : Class.owner .name
	VLNV [name]	Component.name
	busInterfaces.businterface {BusInterface}	{Port}
BusInterface	**IPXACT!BusInterface**	**UML!Port<<BusInterface>>**
	name	Port.name
	busType	<<BusInterface>>.busType <= Ref <<BusDefinition>> [VLNV]
	abstractionType	<<BusInterface>>.abstractionType <= Ref <<abstractionDefinition>> [VLNV]
	portmapTypes.portType { PortmapType } • portMapType • logicalPort.name • physicalPort.name	portMaps <= { String } • String s = “(logicalPort’=%logicalport.name%’,physicalPort=’% physicalPort.name%’)”

	One is not null among(master ,slave , system, mirroredMaster, mirroredSlave, mirroredSystem)	
	master (*master != null*)	<<BusInterface>>.interfaceMode= InterfaceModeKind.master
	slave (*slave != null*)	<<BusInterface>>.interfaceMode= InterfaceModeKind.slave
	system (*system != null*)	<<BusInterface>>.interfaceMode= InterfaceModeKind.system
	mirroredMaster (*mirroredMaster != null*)	<<BusInterface>>.interfaceMode= InterfaceModeKind.mirroredMaster
	mirroredSlave (*mirroredSlave != null*)	<<BusInterface>>.interfaceMode= InterfaceModeKind.mirroredSlave
	mirroredSystem (*mirroredSystem != null*)	<<BusInterface>>.interfaceMode= InterfaceModeKind.mirroredSystem
	all are null	<<BusInterface>>.interfaceMode= InterfaceModeKind.none

BusDefinition	**BusDefinition [VLNV]**	**Class <<BusDefinition>>**
		Package: Class.owner.owner.name= "busdefintion"
	VLNV [libray]	Package: Class.owner.name
	VLNV [name]	Class.name
	VLNV [vendor, version]	<<BusDefinition>>.ident ="vendor='%vendor%' , version='%version%' ")
	extends (if extends *!=null)*	<<BusDefinition>>.extends <= Ref <<Busdefintion>> [VLNV]

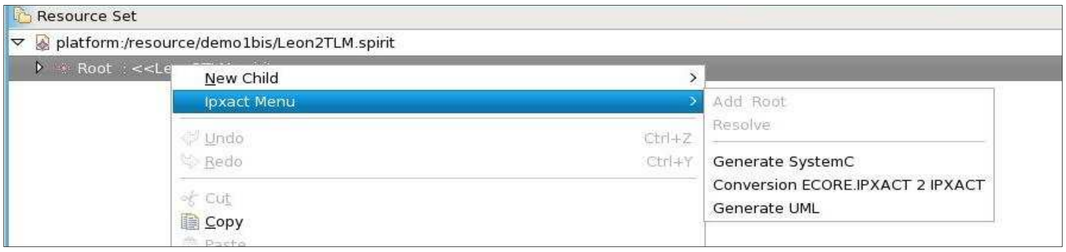
AbstractionDefinition	**AbstractionDefinition [VLNV]**	**Class <<AbstractionDefinition>>**
		Package:Class.owner.owner.name "abstractionDefinition"
	VLNV [libray]	Package:Class.owner.name
	VLNV [vendor, version]	<<AbstractionDefinition>>.ident ="vendor='%vendor%' , version='%version%' ")
	extends (if *extends !=null*)	<<AbstractionDefinition>>.extends <= Ref AbstractionDefinition [VLNV]
	busType	<<AbstractionDefinition>>.busType <=Ref Busdefintion [VLNV]
	portType: 2 cas • Transactional • Wire	Class <<..>> is child of Class <<**AbstractionDefinition**>>
Transactional	Transactional (portType.transactional *!=null)*	Class <<AbsTransctionnalPort>>
	transactional.qualifier.isAddress	<<AbsTransactionalPort>>.isAddress
	transactional.qualifier.isData	<<AbsTransactionalPort>>.isData
	transactional.onMaster.presence	<<AbsTransactionalPort>>.PresenceOnMaster
	transactional.onMaster.service.initiative	<<AbsTransactionalPort>>.InitiativeOnMaster
	transactional.onMaster.service.typeName.value	<<AbsTransactionalPort>>.ServicetypeOnMaster
	transactional.onMaster.service.typeName.implicit	<<AbsTransactionalPort>>.ServicetypeImplicitOnMaster
	transactional.onSlave.presence	<<AbsTransactionalPort>>.PresenceOnSlave
	transactional.onSlave.service.initiative	<<AbsTransactionalPort>>.InitiativeOnSlave
	transactional.onSlave.service.typeName.value	<<AbsTransactionalPort>>.ServicetypeOnSlave
	transactional.onSlave.service.typeName.implicit	<<AbsTransactionalPort>>.ServicetypeImplicitOnSlave

Wire	Wire (*portType.wire !=null*)	Class <<AbsWirePort>>
	wire.qualifier.isAddress	<<AbsWirePort>>.isAddress
	wire.qualifier.isData	<<AbsWirePort>>.isData
	wire.qualifier.isClock	<<AbsWirePort>>.isClock
	wire.qualifier.isReset	<<AbsWirePort>>.isReset
	wire.onMaster.presence	<<AbsWirePort>>.PresenceOnMaster
	wire.onSlave.presence	<<AbsWirePort>>.PresenceOnSlave

1 Launch Transformation:

• In Ecore.IPXACT editor

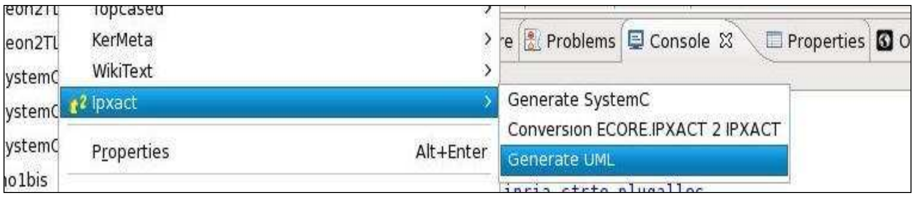
Select an element in model, and in its context menu, use a command: **Ipxact Menu >> Generate UML**



Caution: you export the current document with the latest changes even if they have not saved.

• In the Workspace

In the Workspace, select file * .spirit, and in its context menu, use a command: **Ipxact >> Generate UML**



for a file *myfichier.spirit* ==> *myfichier.spirit.uml*, if there are Designs in the model and Papyrus is installed then, there will also file *myfichier.spirit.di2*.

6 Transformation FromIPXACT to SystemC

1 IPXACT to SystemC

Consider a design, with components (all with an implementation in SystemC)

For this design we create a file SystemC as:

Design	/* * Generate design : [*VLNV*] * */ #include "systemc.h"
For all the components used, it looks for all the systemC headers	*List of all includes*
	int sc_main(int ac ,char *av[]) {
Creation of signals : Creating a variable so the name is that of the signal	// Declaration des signaux d'interconnection sc_signal<***%signalType%***> ***%signalID%***
Creation of instances : for all ComponentInstance *instance* • *instance*.name ==> %instanceName% • *instance*.conponentRef ==> %instanceType%	// Declaration des Instances ***%instanceType% %instanceName%*** ("***%instanceName%***") ;
Creation of links between the port and the signals for all Interconnection *interconnection* • *interconnection ==> %signalID%* • *interconnection*.activeInterface{ } • componentRef ==> %InstanceName% • busRef ==> %portName%	// Mapping Interconnection ***%instanceName%***.***%portname%***(***%signalID%***) ;

	sc_start();
	return 0;
	}

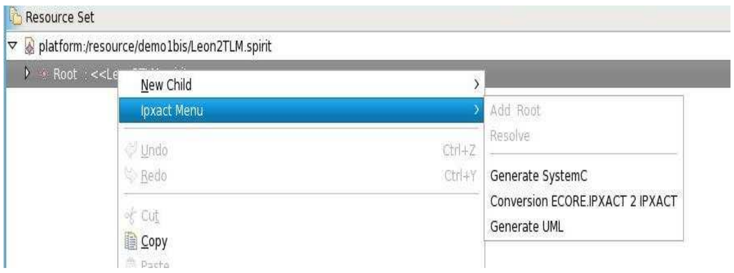
you must complete the file. Check constructors calling are valid.

Warning: if the design is not correct, it may have errors at compile or execution.
Ex: Problem typing between port, Port Non Connected

2 Launch generation

- **In Ecore.IPXACT editor**

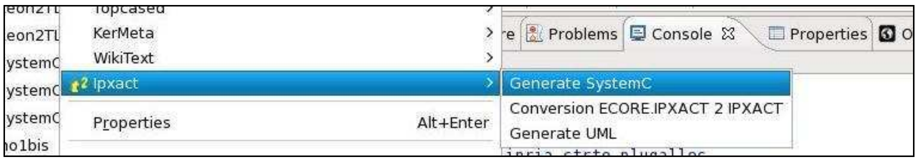
Select an element in model, and in its context menu, use a command: **IPXACT Menu >> Generate SystemC**.



Caution: you export the current document with the latest changes even if they have not saved.
.

- **In the Workspace**

In the Workspace, select file * .spirit, and in its context menu, use a command: IPXACT >> **Generate SystemC**.



In the directory the source file, there is a subdirectory "outputSystemC" in which there are files *. cc (one file per design)