Kactus2: Hierarchical design tutorial

Introduction

This tutorial describes how to design a hierarchical system using Kactus2 software. This tutorial covers the creation of IP blocks, connecting IPs together using buses, creating hierarchical system using existing IPs, creating system design (SW/HW mapping) and automatic creation of documentation from the designed system.

Used library available at http://opencores.org/project,funbase_ip_library (Requires registration) or at course web pages.

Kactus2 tool can be downloaded from http://sourceforge.net/projects/kactus2/

For more information about the Kactus2 software, refer to http://funbase.cs.tut.fi.

Requirements

Kactus2 Software version 2.0 (Build 1) or newer

Revision History

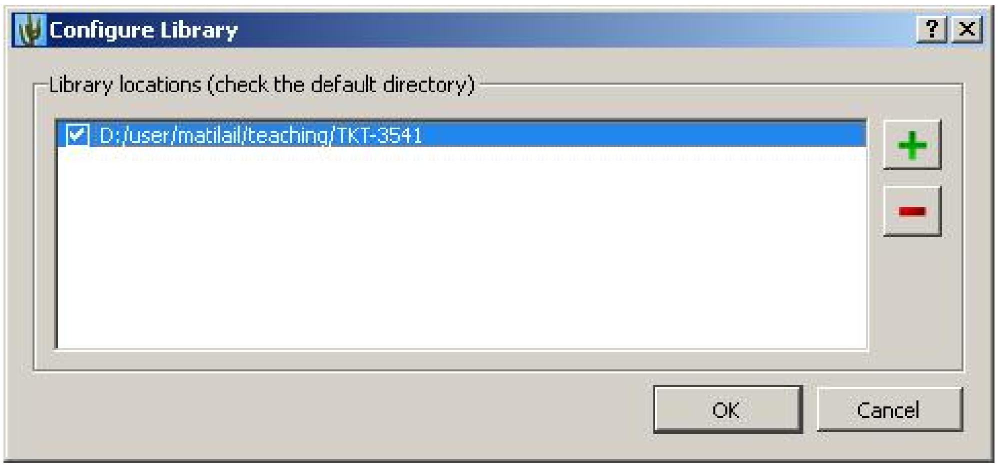
Version	Date	Author	Comments
1.0	11.01.2012	Jussi Raasakka	Initial version
1.1	12.01.2012	Jussi Raasakka	Changed VLNV naming convention to follow funbase naming convention
1.2	13.01.2012	Jussi Raasakka	Corrections for VLNV naming conventions
2.0	08.01.2013	Lauri Matilainen	Updated instructions to use Kactus2 v2.0

Table of Contents

Configure library paths

Open the library configuration dialog using the “Configure library” icon found from the Library group in the top pane.

Add the default directory to point on the network disk where you downloaded the library from the opencores.org to make sure that you can access created files from different computers (see Figure 1). You may need to set this setting every time you start the Kactus2 software from different computer.



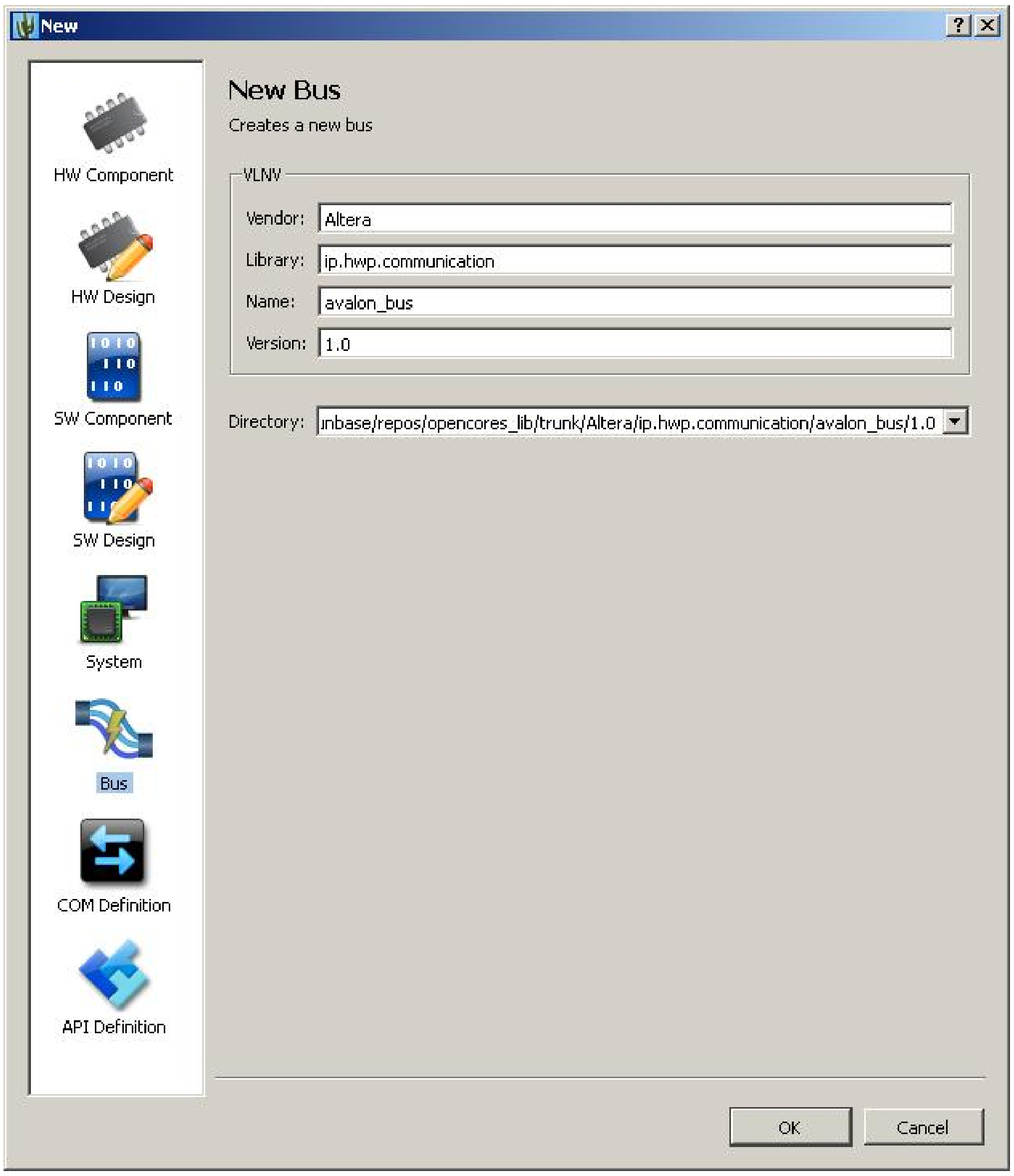
☑ D:/user/matilail/teaching/TKT-3541

Figure 1. Kactus2 library configuration

Creating a bus

This bus is pre-given in the exercise work library.

In the top pane, under File group select New to create new bus. After new window has appeared, select Bus from the left hand menu. On the right side you should see several VLNV fields which are shown in Figure 2.



New Bus

Creates a new bus

VLNV

Vendor: Altera

Library: ip.hwp.communication

Name: avalon_bus

Version: 1.0

Directory: ınbase/repos/opencores_lib/trunk/Altera/ip.hwp.communication/avalon_bus/1.0

Figure 2. Creating a new bus.

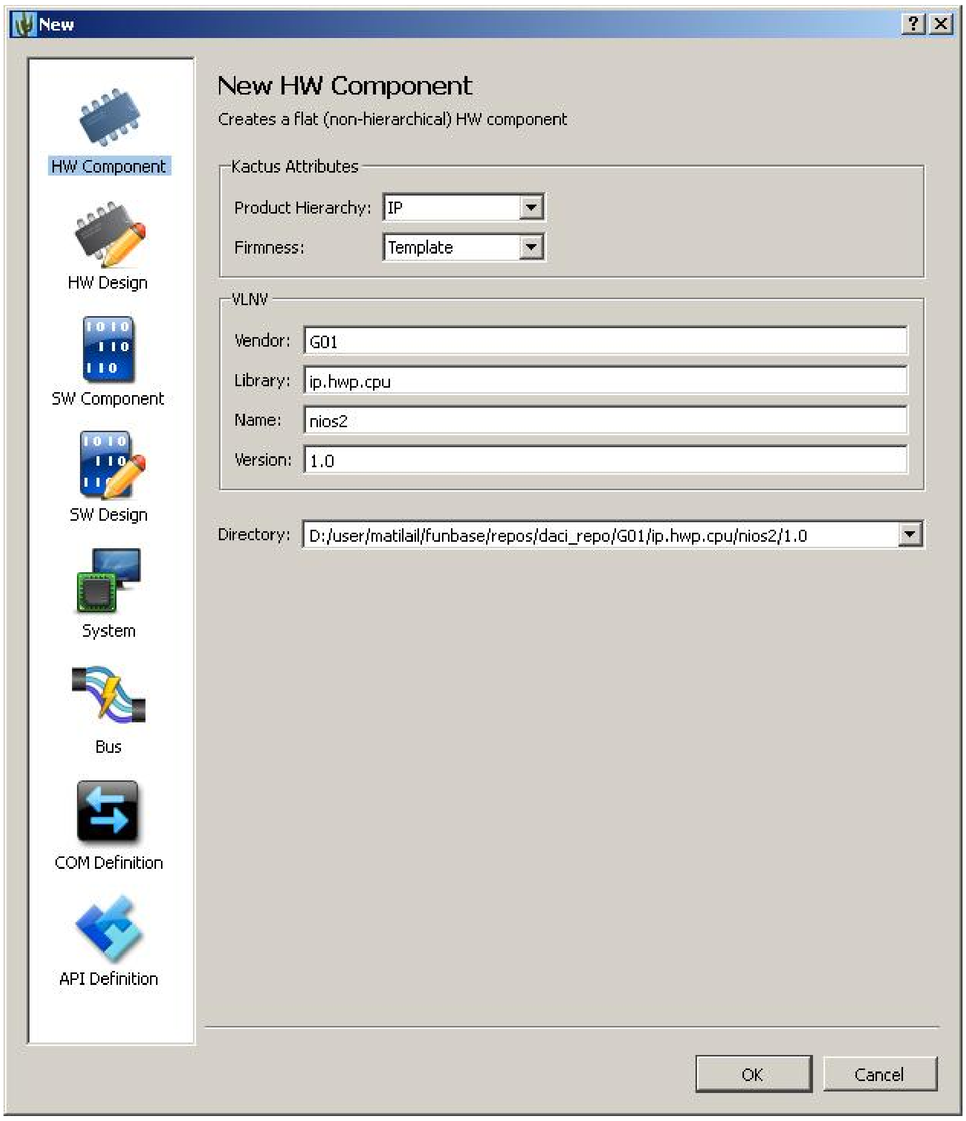
IP-XACT format requires you to give each object a unique identifier defined by Vendor, Library, Name, and Version fields. In this tutorial we are using the naming convention specified in [1] Set Vendor to "Altera". Use word "ip.hwp.communication" for the Library field, "avalon_bus" for the Name field, and "1.0" for the Version field. After writing the VLNV fields, press OK to create the bus.

After creating the bus we would normally define the bus signals. In this tutorial we are focusing on creating the hierarchical system and documentation features of the Kactus2 software, so for now we can create buses without any signals. You can now close the appeared tab "avalon (1.0) [bus]" tab.

Creating a HW component

Creating new HW component follows the same principles as creating a new bus. First select new from the top pane, under File group. From the pop-up window select HW Component from the left hand menu. Select IP from the Product pull down menu and Template from the Firmness pull down menu.

The Product pull down option lets you choose which type of component you are specifying. Firmness option gives information how the component can be modified.



New HW Component

Creates a flat (non-hierarchical) HW component

Kactus Attributes

Product Hierarchy: IP

Firmness: Template

VLNV

Vendor: G01

Library: ip.hwp.cpu

Name: nios2

Version: 1.0

Directory: D:/user/matilail/funbase/repos/daci_repo/G01/ip.hwp.cpu/nios2/1.0

Figure 3. Creating a HW new component.

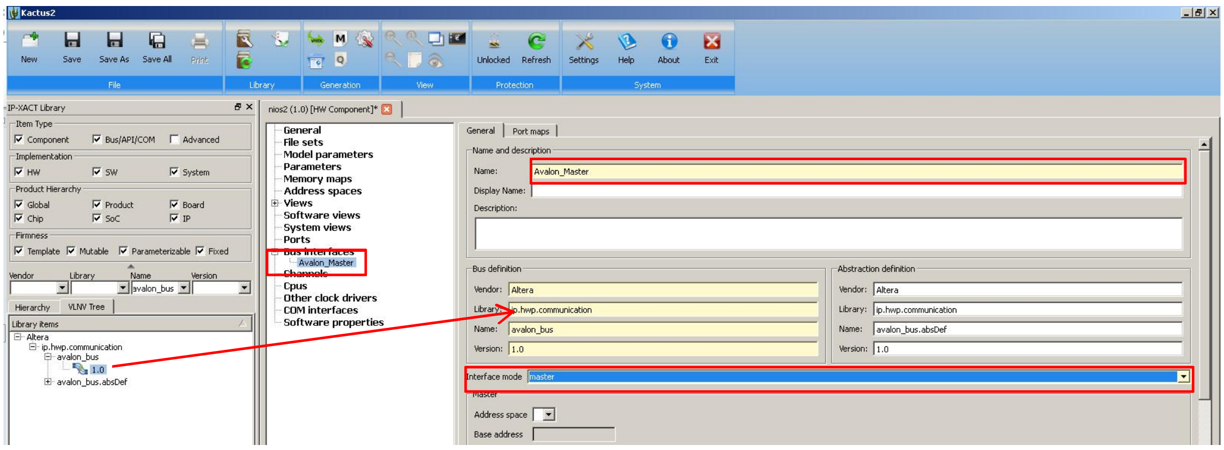
Fill the VLNV tuple with your G<group number> (group 3 would use G03 etc…)., “ip.hwp.cpu”, “nios2”, and “1.0” (see Figure 3). Press OK to create the component.

Adding interfaces to components

After creating new component, new tab presenting the component opens which should be filled with information about the component. In this tutorial we are only focusing on creating components, buses and connections between components. So for this tutorial the only information that needs to be filled for each component are the bus interfaces and CPU information.

For Nios2 processor component we only need “avalon_bus” bus. Select the “Bus interfaces” from the list shown on the nios2 tab and double click on the empty list space to create new bus interface. In the name field type “Avalon_Master”. In addition you can give Display name and Description to this interface if you wish. NOTE: You are not able to edit Bus Definition column in the Bus interfaces list. Open the Bus interface by double clicking “Avalon Master” interface from the tree view Bus Interfaces->Avalon Master.

To tie this bus interface to the actual bus we need to fill in the Bus definition and Abstraction definition fields. These can be easily filled by using drag and drop method from the hierarchy pane. Expand the avalon_bus bus from the hierarchy pane to show the Abstraction definition for this bus.The resulting bus interface definition should look the same as in Figure 4.



Item Type: ☑ Component ☑ Bus/API/COM ☐ Advanced
Implementation: ☑ HW ☑ SW ☑ System
Product Hierarchy: ☑ Global ☑ Product ☑ Board ☑ Chip ☑ SoC ☑ IP
Firmness: ☑ Template ☑ Mutable ☑ Parameterizable ☑ Fixed
Vendor: Library: Name: avalon_bus Version:

- Altera
  - ip.hwp.communication
    - avalon_bus
      - 1.0
    - avalon_bus.absDef

Name and description
Name: Avalon_Master
Display Name:
Description:

Bus definition
Vendor: Altera
Library: ip.hwp.communication
Name: avalon_bus
Version: 1.0

Abstraction definition
Vendor: Altera
Library: ip.hwp.communication
Name: avalon_bus.absDef
Version: 1.0

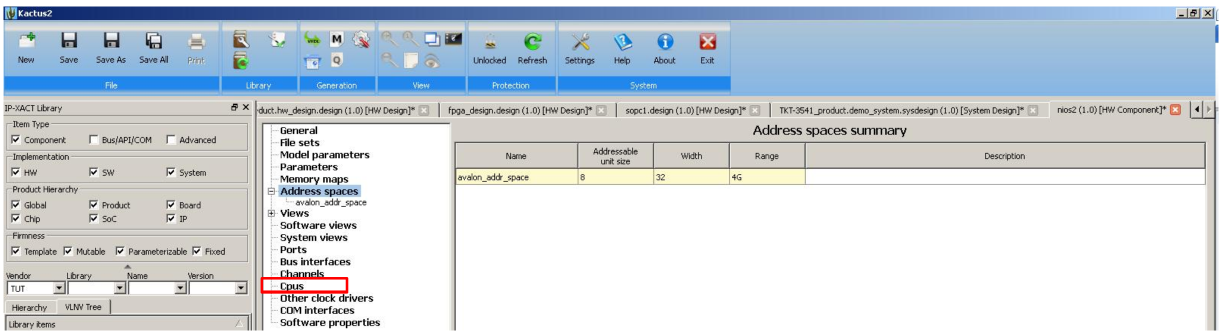
Interface mode: master

Master
Address space
Base address

Figure 4. Defining bus interface for a component.

The only remaining thing left to specify is the interfacing mode for the bus interface. The most typical settings here are either master or slave mode. This setting can be found from the Interface mode section. As the Nios2 processor acts as a master on the avalon_bus bus, select master from the pull down menu shown in **Error! Reference source not found.**. More bus interfaces can be added to if required using the same basic principle.

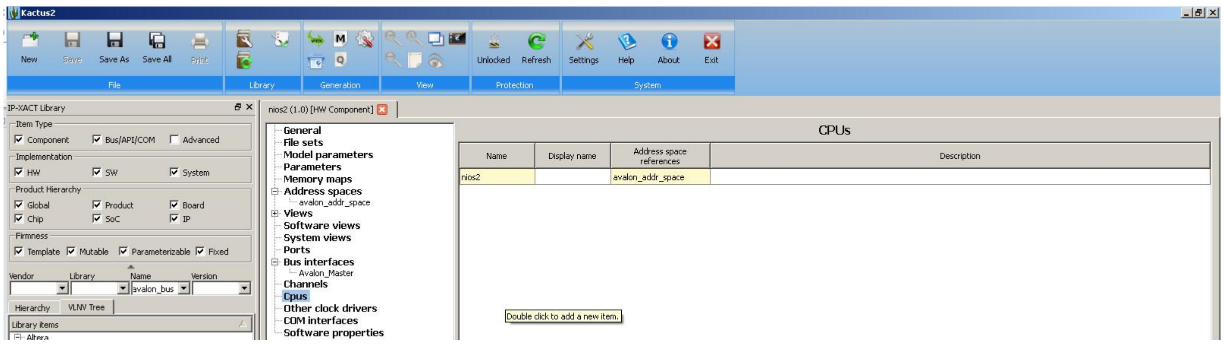
To identify created component as a CPU component we have to add address space to the component. Select Address spaces from the left list. Add new address space to address space summary list and fill the cells as shown in Figure 5. After that you are able identify the component as a programmable one. Select CPUs and add new line to CPUs list by double clicking empty area. Name it as you wish and select address space reference to created avalon_addr_space. Save component.



Address spaces summary

Name	Addressable unit size	Width	Range	Description
avalon_addr_space	8	32	4G	

Figure 5 Address space addition



CPUs

Name	Display name	Address space references	Description
nios2		avalon_addr_space	

Double click to add a new item.

Figure 6 CPU definition

Creating a channel component

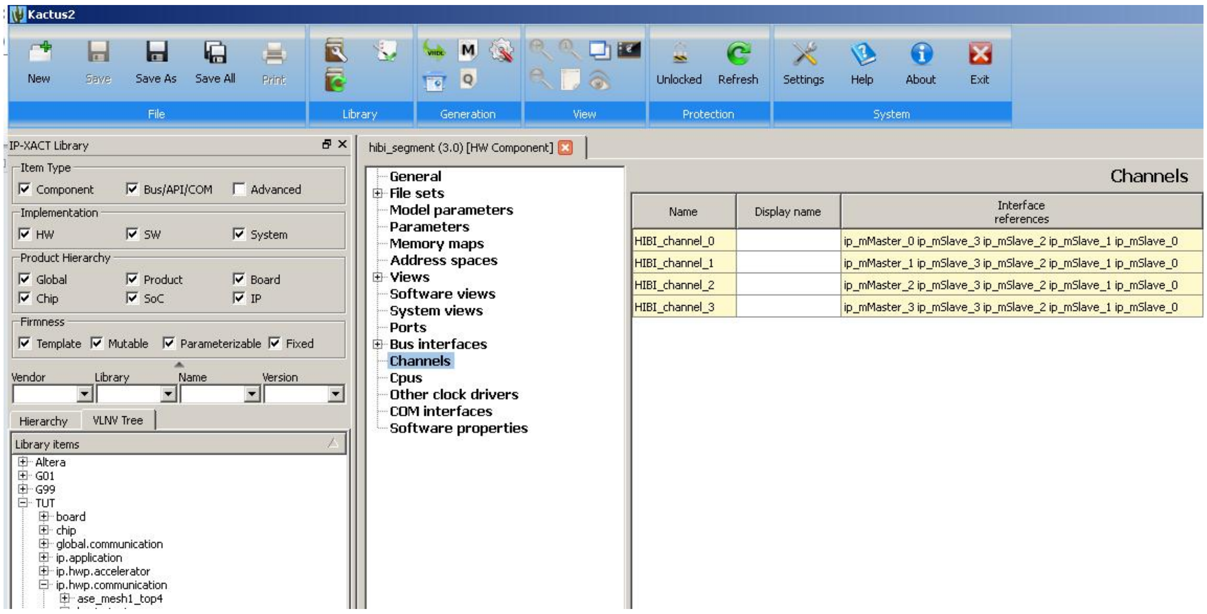
This component is pre-given in the exercise work library.

To make the HIBI bus we need to make special component that represents the HIBI interconnection network. Start by creating normal component with Product hierarchy “IP”, and fill the VLNV tuple as G<group number>, “ip.hwp.communication”,“hibi”, and “draft”. Add following bus interfaces to the component

- Slave interface to system_clk_bus
- Slave interface to rst_n_bus
- Mirrored master interface to hibi_bus
- 2nd mirrored master interface to hibi_bus

- 3rd mirrored master interface to hibi_bus

After creating the required bus interfaces, select the “Channels” from the list shown on the nios2 tab and press + to create new channel. Type “hibi_Channel” to the name field and add all hibi_bus bus interfaces to the channel by double clicking in the bus interface references field and selecting the right bus interfaces. After adding all bus interfaces the result should look similar to Figure 7.



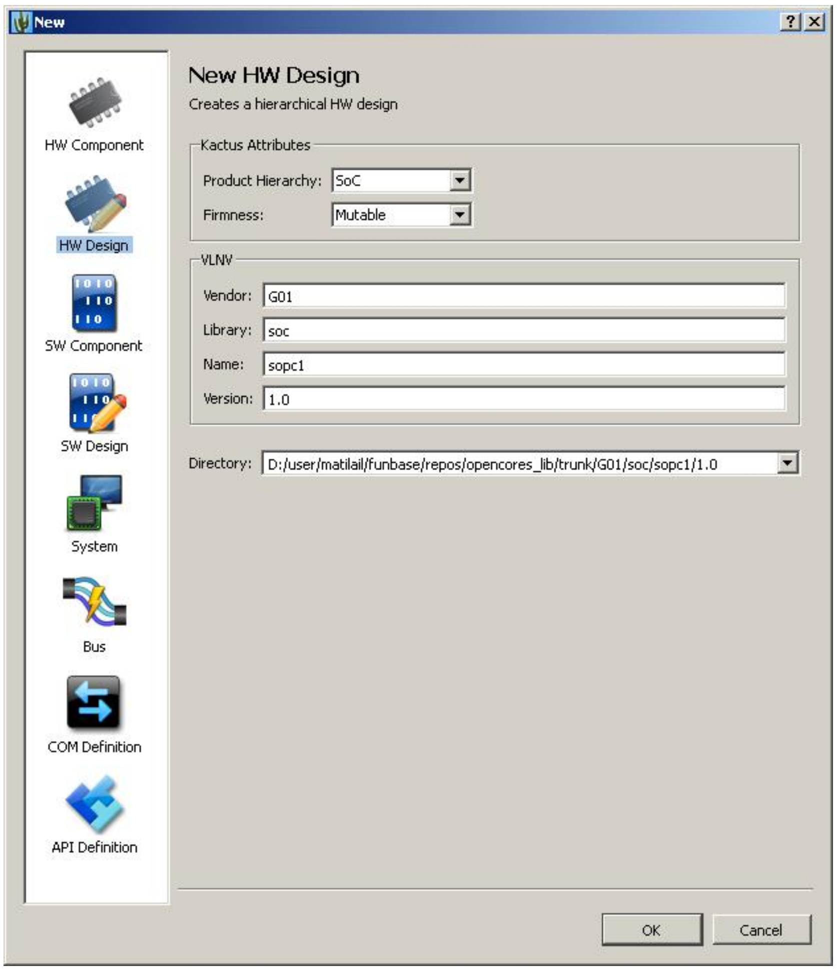
Name	Display name	Interface references
HIBI_channel_0		ip_mMaster_0 ip_mSlave_3 ip_mSlave_2 ip_mSlave_1 ip_mSlave_0
HIBI_channel_1		ip_mMaster_1 ip_mSlave_3 ip_mSlave_2 ip_mSlave_1 ip_mSlave_0
HIBI_channel_2		ip_mMaster_2 ip_mSlave_3 ip_mSlave_2 ip_mSlave_1 ip_mSlave_0
HIBI_channel_3		ip_mMaster_3 ip_mSlave_3 ip_mSlave_2 ip_mSlave_1 ip_mSlave_0

Figure 7. Creating a channel.

Creating hierarchical design

Now that all the components and buses are made for this tutorial, we can proceed to create the actual hierarchical system.

Select new from the top pane, under File group. From the pop-up window select HW Design from the left hand menu. Select SoC from the Product Hierarchy pull down menu and correct Firmness to your SoC design. Fill the VLNV tuple. The end result should look similar to Figure 8. Press OK to create component.



New HW Design

Creates a hierarchical HW design

Kactus Attributes

Product Hierarchy: SoC

Firmness: Mutable

VLNV

Vendor: G01

Library: soc

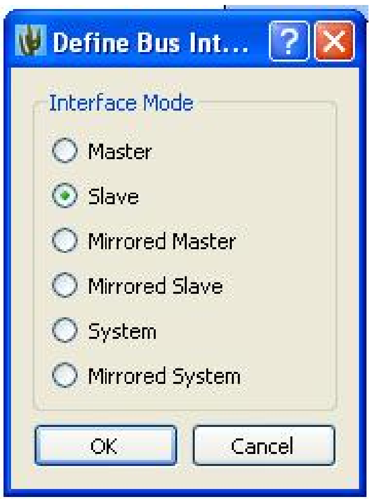
Name: sopc1

Version: 1.0

Directory: D:/user/matilail/funbase/repos/opencores_lib/trunk/G01/soc/sopc1/1.0

Figure 8. Creating a new design.

Now we can start adding components and interconnections between components. You can drag and drop different components to the design using the hierarchy or VLNV tree menu on the left pane. Creating ports and connecting components together with buses can be done via the tools provided in the Diagram Tools section in the top pane. Add **nios2**, **sdram_ctrl**, **avalon_onchip** and **hibi_pe_dma** components to the design, make two input and output ports to the design. The two input ports are reserved for the clk and rst_n buses. We are not going to connect them to the components, but we can specify the input ports to be buses clk_bus and rst_n_bus by dragging and dropping the clk_bus **abstraction definition** and rst_n_bus **abstraction definition** over the input ports. When dragging select slave from the pop-up menu (see Figure 9). After connecting the bus definition to the port you must give it a name by clicking the port and typing it name from the right pane shown in Figure 10. Type in names “system_clk” and “rst_n”. Connect rest of the interfaces together as shown in the Figure 10. When making connections from the bus interface to the port Kactus2 asks whether you want to auto-copy port maps or setup them manually (see Figure 11). Select Copy to automatically copy port maps.


Define Bus Int...

Interface Mode

- Master
- [✓ Slave]
- Mirrored Master
- Mirrored Slave
- System
- Mirrored System

OK Cancel

Figure 9. Pop-up menu when connecting bus to a port.

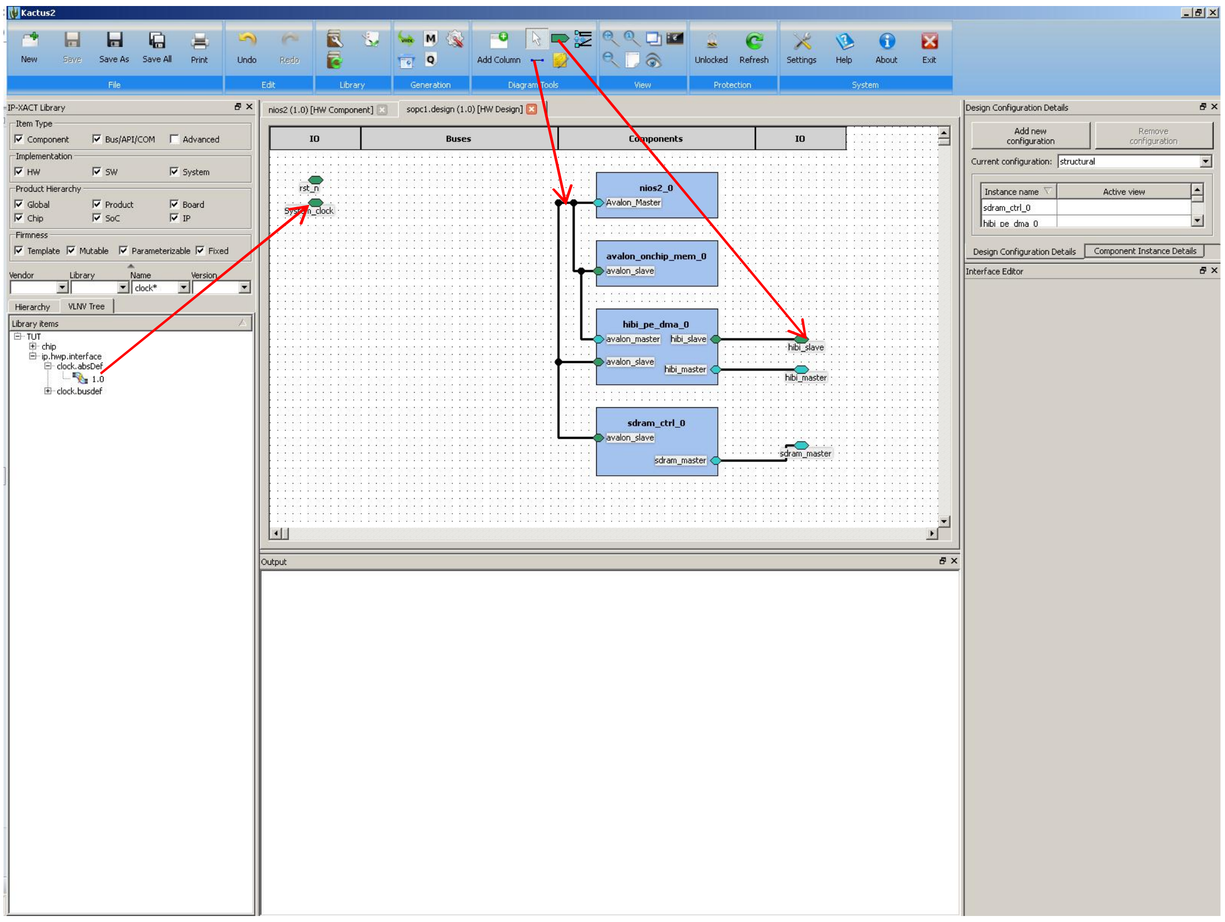
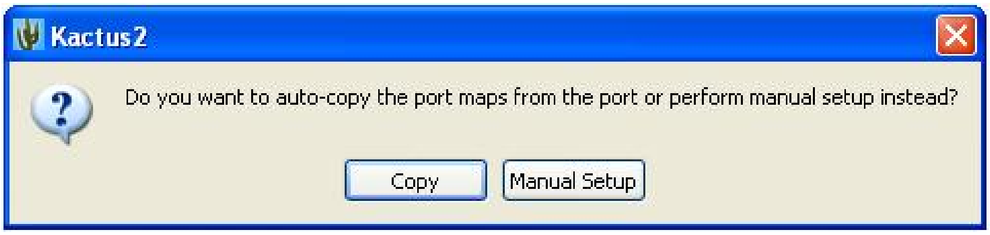


Figure 10. Creating hierarchical design.

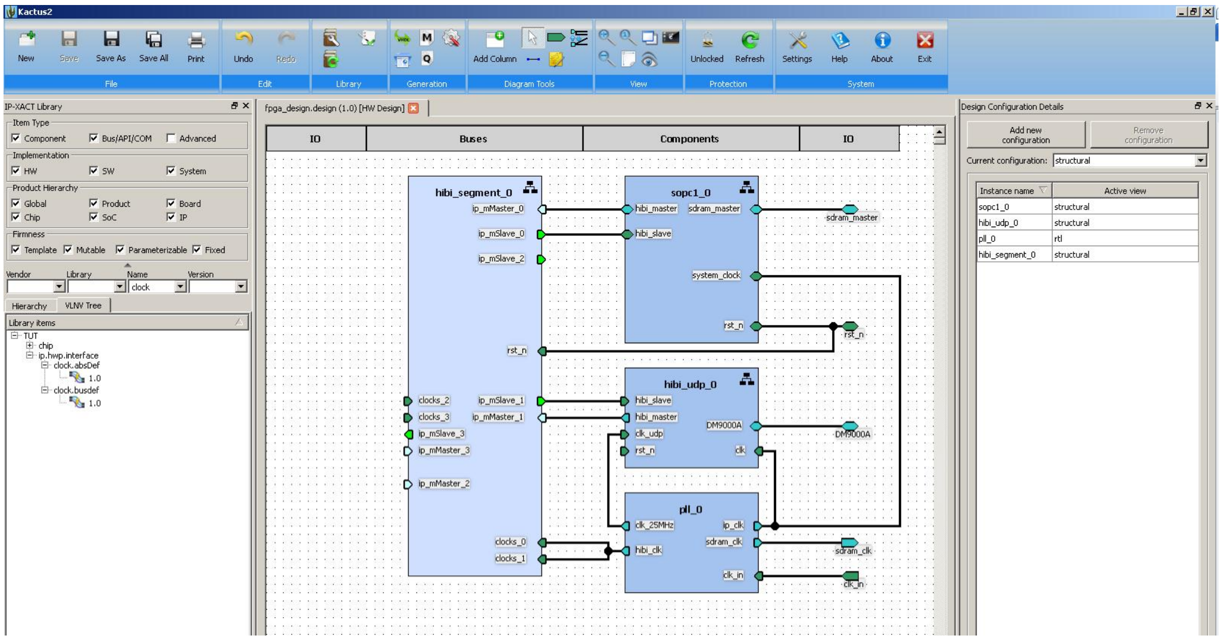

Kactus2

Do you want to auto-copy the port maps from the port or perform manual setup instead?

Copy | Manual Setup

Figure 11. Connecting bus to a port.

Now that we have the two SoPC builder system ready we can move to one level up in the hierarchy. Create another design using the principles shown before. Select “SoC” from the product Hierarchy pull-down menu and fill the the VLNV tuple with your G<group number>, ”soc”, “FPGA_design”, and “1.0”. Connect all the components as shown in Figure 12. You can create new columns for components and IO, by selecting add column from the top pane under diagram tools. Ordering of the columns can be changed by simple drag and drop operation. If you want to be able to browse through hierarchy levels by double clicking the components, select structural as the active view for hierarchical components (sopc1). This setting can be found from the right side pane (see Figure 12). Resulting FPGA_Design should be similar to Figure 12.



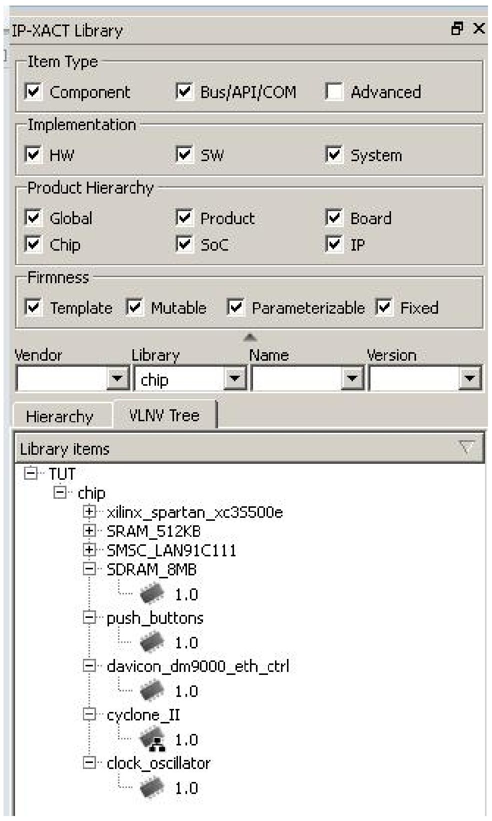
Current configuration: structural

Instance name	Active view
sopc1_0	structural
hibi_udp_0	structural
pll_0	rtl
hibi_segment_0	structural

Figure 12. FPGA Design.

Now we have created the the SoC presenting the FPGA design of our system. Next task is to create board level design, where we model the Altera DE2 development board.

Create another design with product hierarchy “board” and the VLVN tuple as G<group number>, “board”, “de2” and “1.0”. Add and connect components as shown in Figure 13. HINT: set Library dialer in the IP-XACT Library window to chip



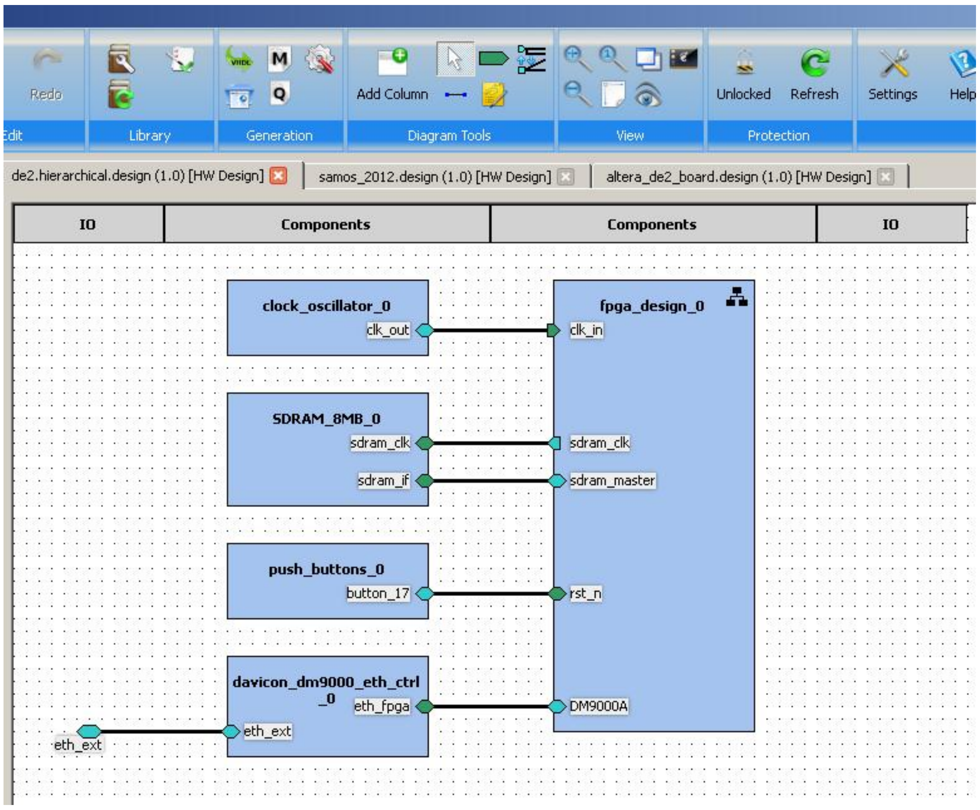


Figure 13. Board level design.

Finally we are able to create highest HW hierarchy level where we connect PC board and de2 board. Create new HW design as previously presented and select Product hierarchy level.

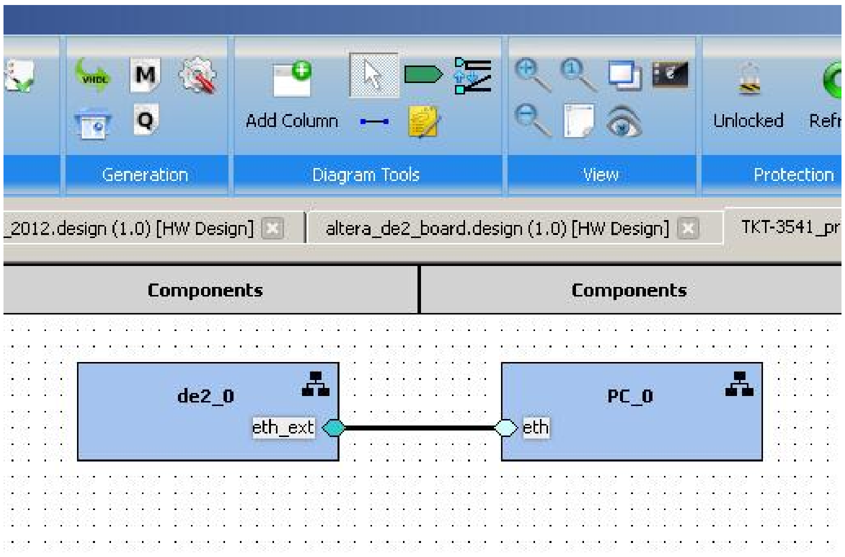
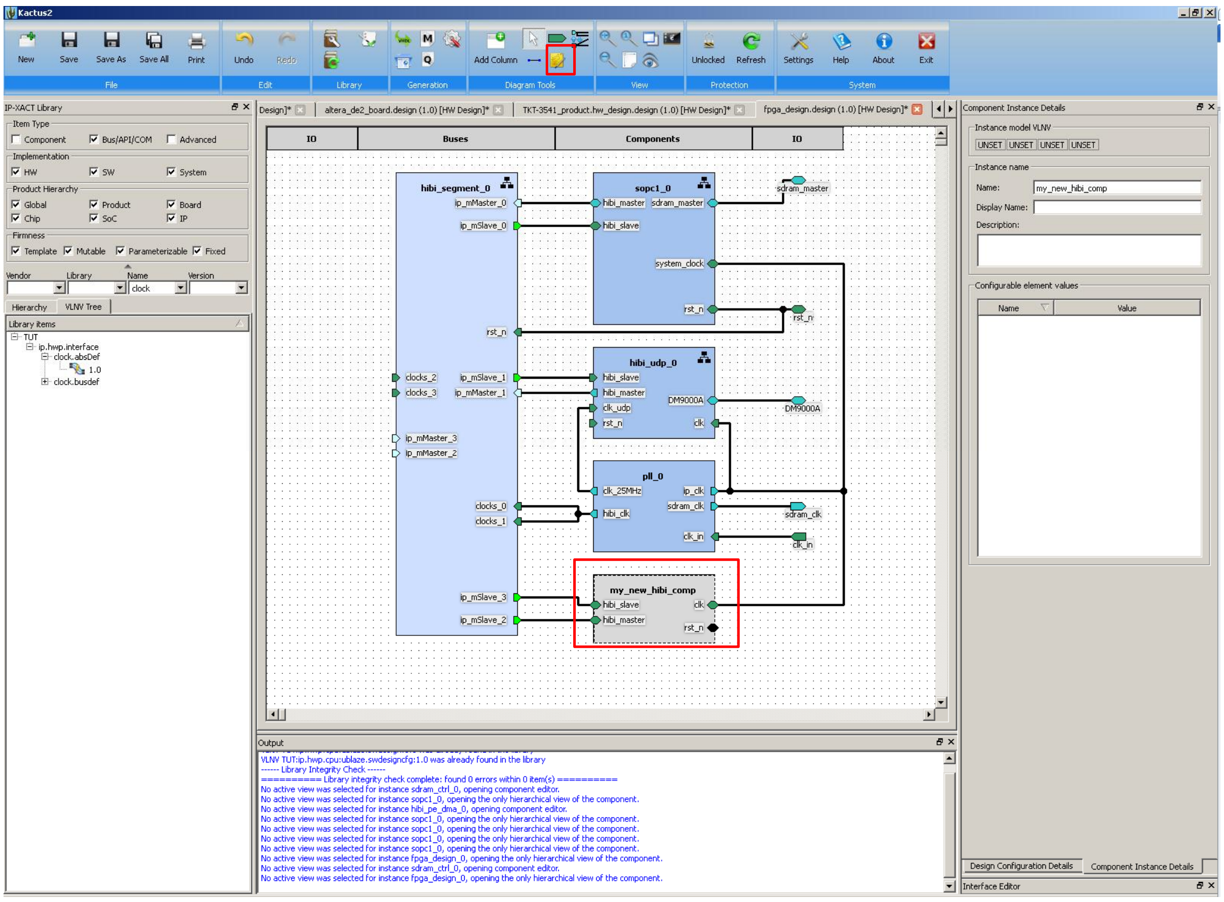


Figure 14 Product level design

Draft tool

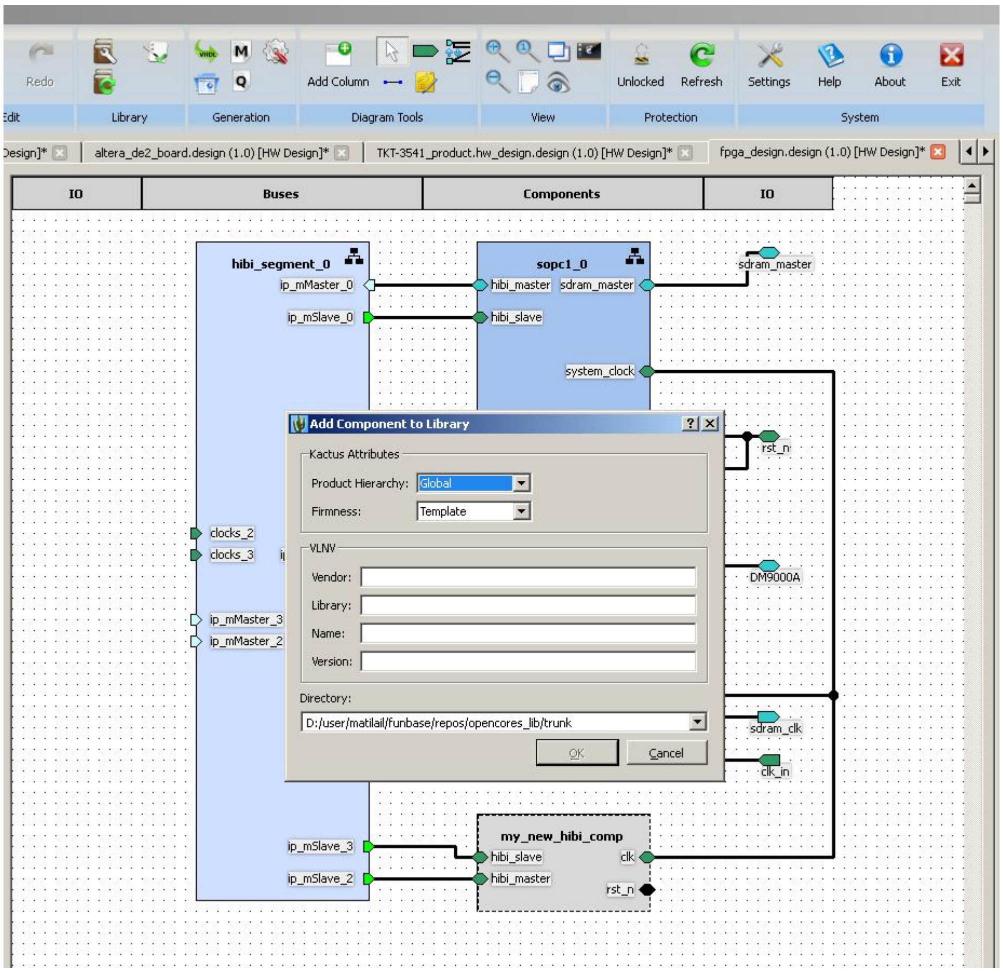
Drafting is easy way to add a new component to the design without creating a new component. You can add a new draft component to the design by selecting Drafting tool from the Diagram tool section and clicking in the design area. Depending which column you click the corresponding draft object appears to the column. E.g. by clicking in the IO-column it creates a draft interface and in the Component column it creates a draft component. You can add interfaces to draft component by clicking the component with Drafting tool. You are able to name draft component and interfaces inside as it is normal component. When you want to package component and add it to the library you can do it by double clicking draft component. NOTE: Do not bother the error messages you get from the draft components in designs.



Name: my_new_hibi_comp

```
VLNV TUT:ip.hwp.cpu:ublaze.swdesigncfg:1.0 was already found in the library
------ Library Integrity Check ------
========== Library integrity check complete: found 0 errors within 0 item(s) ==========
No active view was selected for instance sdram_ctrl_0, opening component editor.
No active view was selected for instance sopc1_0, opening the only hierarchical view of the component.
No active view was selected for instance hibi_pe_dma_0, opening component editor.
No active view was selected for instance sopc1_0, opening the only hierarchical view of the component.
No active view was selected for instance sopc1_0, opening the only hierarchical view of the component.
No active view was selected for instance sopc1_0, opening the only hierarchical view of the component.
No active view was selected for instance sopc1_0, opening the only hierarchical view of the component.
No active view was selected for instance fpga_design_0, opening the only hierarchical view of the component.
No active view was selected for instance sdram_ctrl_0, opening component editor.
No active view was selected for instance fpga_design_0, opening the only hierarchical view of the component.
```

Figure 15 Add new components to the design with drafting tool



Kactus Attributes

Product Hierarchy: Global

Firmness: Template

VLNV

Vendor:

Library:

Name:

Version:

Directory:

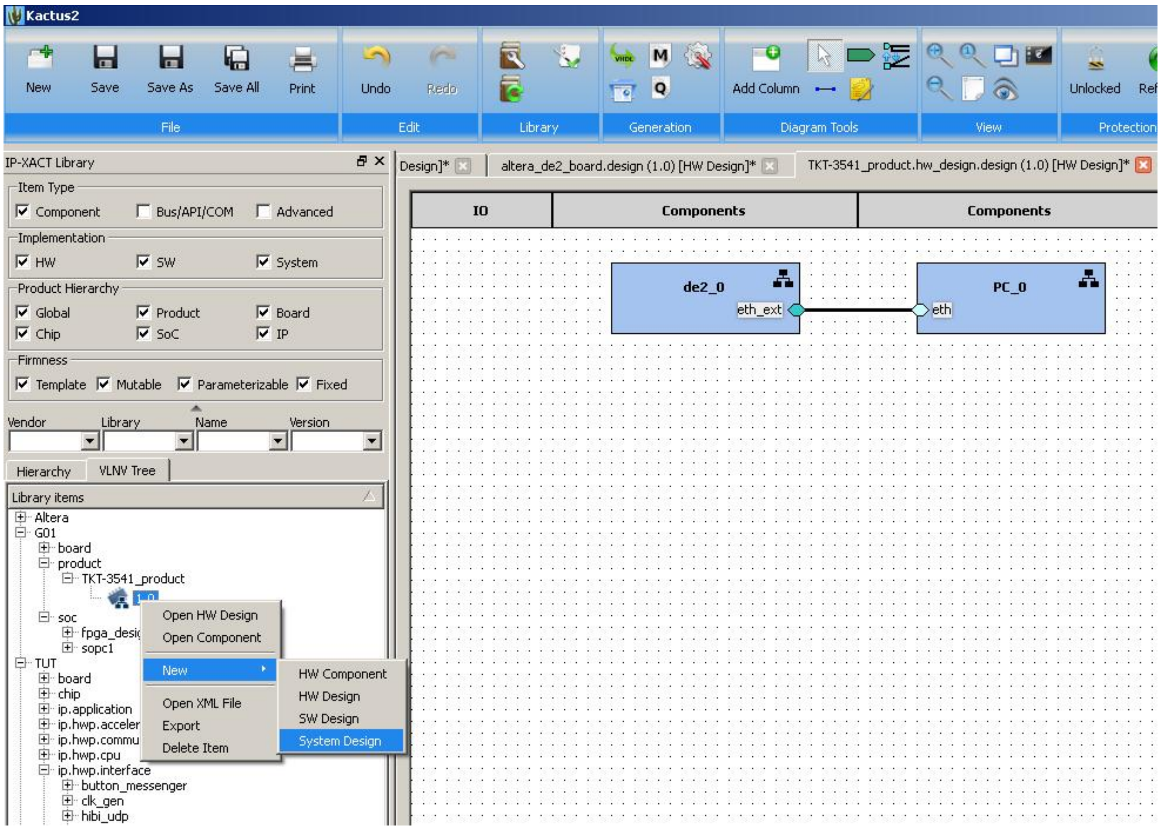
D:/user/matilail/funbase/repos/opencores_lib/trunk

Figure 16 Convert draft component to actual IP-XACT component

Creating system design

System design maps SW components to the programmable HW components i.e. CPUs.

Create new system design to previously created product. Kactus2 parses through the selected product and finds all programmable components in the system design. Now you can start to allocate SW components to the processors. Easy way to add new SW components to the design is with drafting tool as demonstrated earlier.



Item Type: [✓ Component] Bus/API/COM Advanced

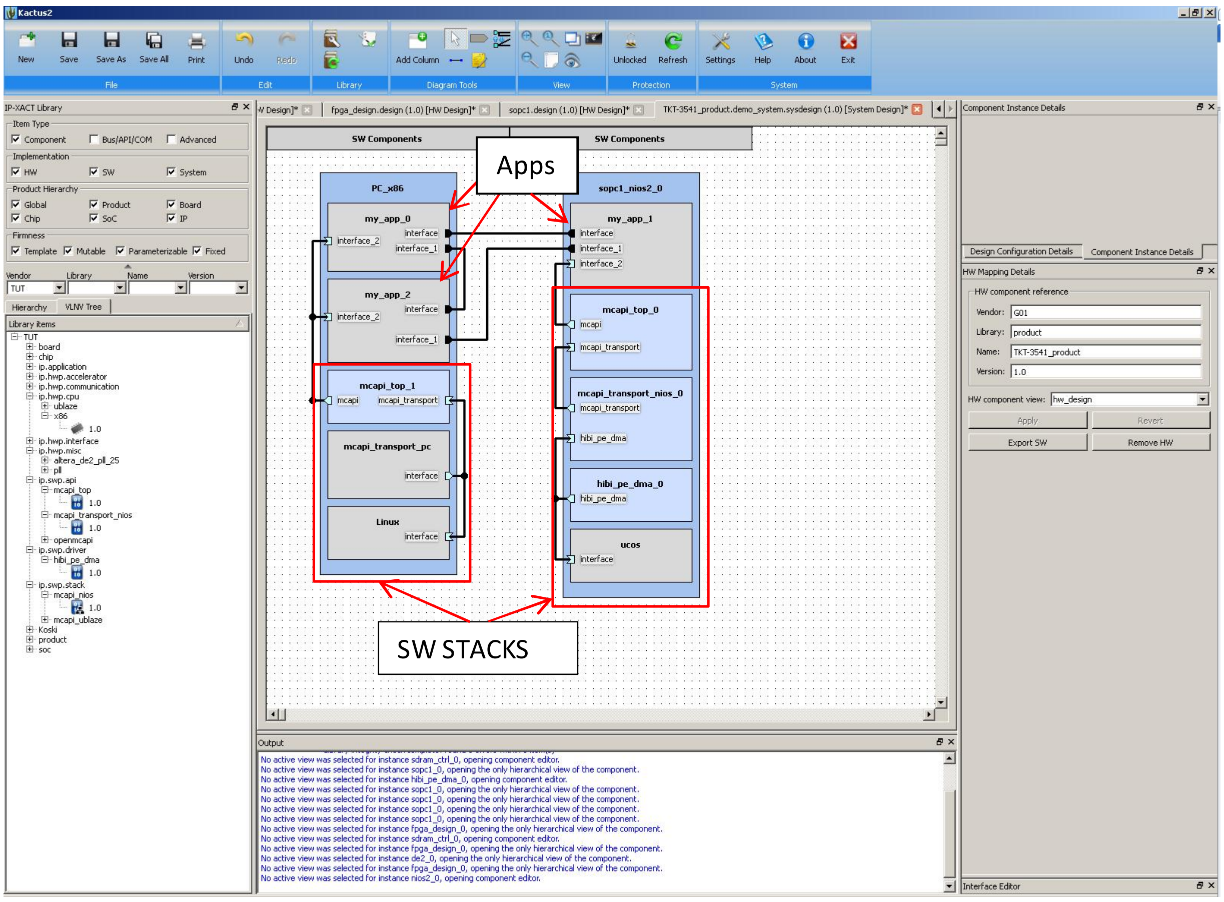
Implementation: [✓ HW] [✓ SW] [✓ System]

Product Hierarchy: [✓ Global] [✓ Product] [✓ Board] [✓ Chip] [✓ SoC] [✓ IP]

Firmness: [✓ Template] [✓ Mutable] [✓ Parameterizable] [✓ Fixed]

Figure 17 System design creation

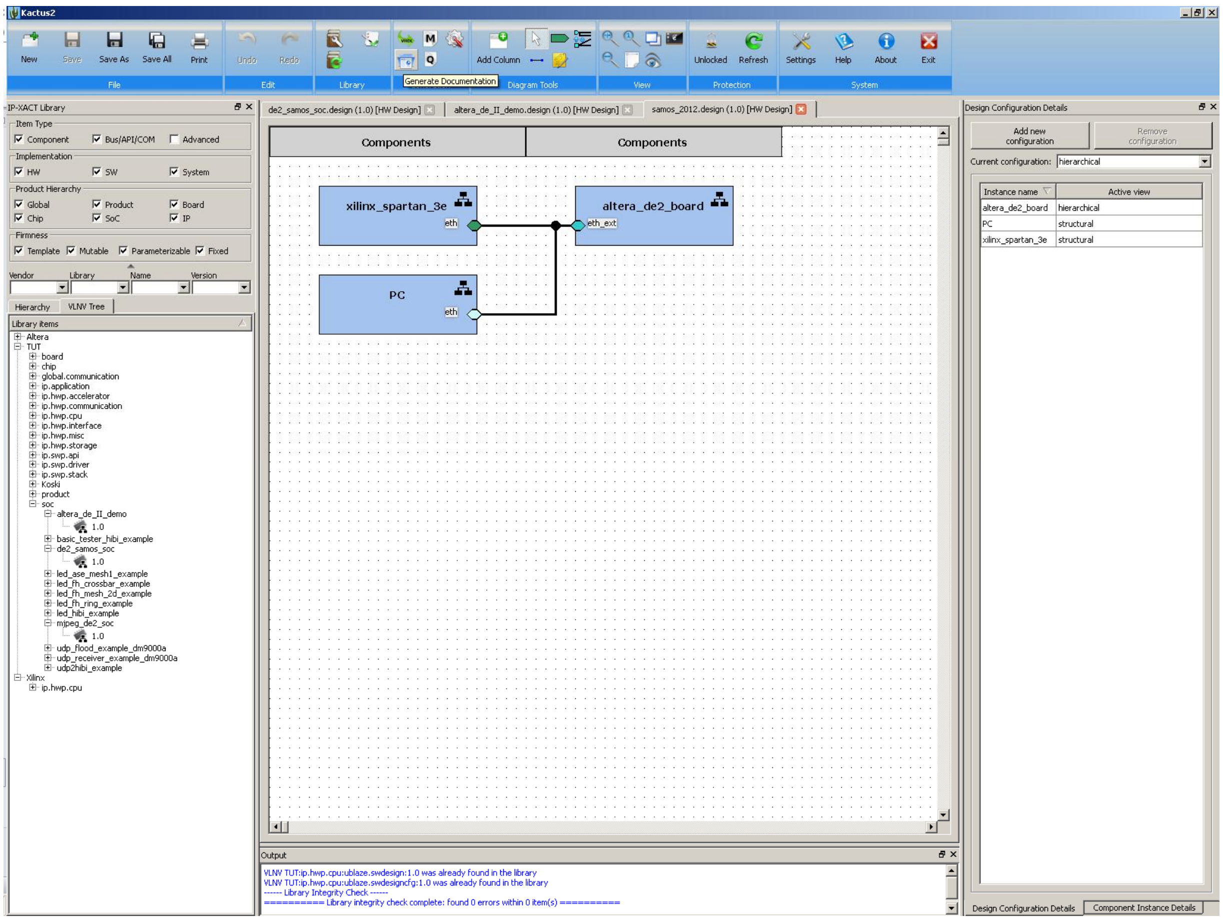
Create SW stacks to processors by drag dropping from the library and add your own components with drafting tool. SW components have interfaces which are used to express APIs between SW stack components and COM interfaces to describe communication between application nodes. NOTE: drafting tool does not differentiate interface types.



```
No active view was selected for instance sdram_ctrl_0, opening component editor.
No active view was selected for instance sopc1_0, opening the only hierarchical view of the component.
No active view was selected for instance hibi_pe_dma_0, opening component editor.
No active view was selected for instance sopc1_0, opening the only hierarchical view of the component.
No active view was selected for instance sopc1_0, opening the only hierarchical view of the component.
No active view was selected for instance sopc1_0, opening the only hierarchical view of the component.
No active view was selected for instance sopc1_0, opening the only hierarchical view of the component.
No active view was selected for instance fpga_design_0, opening the only hierarchical view of the component.
No active view was selected for instance sdram_ctrl_0, opening component editor.
No active view was selected for instance fpga_design_0, opening the only hierarchical view of the component.
No active view was selected for instance de2_0, opening the only hierarchical view of the component.
No active view was selected for instance fpga_design_0, opening the only hierarchical view of the component.
No active view was selected for instance nios2_0, opening component editor.
```

Creating documentation using Kactus2

Document creation is automated in the Kactus2 software. To make documentation of the created design, the only thing that is needed is to click on the Generate documentation button located on the top pane under Generation category (see Figure 18).



Current configuration: hierarchical

Instance name	Active view
altera_de2_board	hierarchical
PC	structural
xilinx_spartan_3e	structural

Output

```
VLNV TUT:ip.hwp.cpu:ublaze.swdesign:1.0 was already found in the library
VLNV TUT:ip.hwp.cpu:ublaze.swdesigncfg:1.0 was already found in the library
------ Library Integrity Check ------
========== Library integrity check complete: found 0 errors within 0 item(s) ==========
```

Figure 18. Creating documentation for design.

Summary

If you successfully executed the tutorial you are now able to describe exercise work system with Kactus2. You can use available IP-XACT components, create your own ones or use draft components.

References

[1] KACTUS2_IP-XACT_IEEE1685 compatible design environment for embedded MP-SoC products web edition, 2011, Tampere University of Technology, [Online], Available http://funbase.cs.tut.fi/images/5/53/KACTUS2_IP-XACT_IEEE1685_compatible_design_environment_for_embedded_MP-SoC_products_web_edition.pdf, pages 40.45, accessed Jan12, 2012