PathWave
FPGA 2018

PathWave FPGA Customer Documentation

KEYSIGHT
TECHNOLOGIES

Notice

Table of Contents

PathWave FPGA Customer Documentation

PATHWAVE	**Keysight PathWave FPGA Documentation** Keysight PathWave FPGA is a system-level FPGA development environment that allows you to create and deploy your custom hardware-acceleration directly into instruments.

Key Features

Overview

Getting Started

User's Guide
Release Notes

Working with PathWave FPGA

GUI Overview
Configuring PathWave FPGA
Creating a New Project

Getting Started

This manual contains the following sections:

- Release Notes

Release Notes

This section contains information about previous and current releases.

Release Highlights

This section provides a general overview of each release.

- PathWave FPGA is a graphical environment that provides a way to rapidly develop FPGA designs on Keysight Open FPGA hardware.
- An IP library is provided which includes Logic/Math, Memory, and DSP blocks that can be included in an FPGA design. Vivado IP blocks or custom HDL IP can also be imported and the port interfaces described using IP-XACT 2014.
- PathWave FPGA provides a design flow from schematic to bitfile generation with the press of a button.

For system requirement details, refer System Requirements. For installation steps, refer Installation.

Licensing

- PathWave FPGA requires: a) **version 2018.04** of the EEsof EDA licensing software, b) version **>=2018.04** codewords to run, and c) the licensing server software, ***lmgrd*** and ***agileesofd***, to be upgraded to at least the same versions as what are included in EEsof EDA Licensing software **2018.04**. PathWave FPGA will not start if any of these requirements is not met.
- In the EEsof EDA License Tools version 2018.04, licensing vendor daemon (***agileesofd***) is upgraded to sync up with FlexNet FNP **11.13.1.4** version of FLEX license manager (***lmgrd***). PathWave FPGA installer for the Windows platform will automatically set up these two new license server daemons by default for the local node-locked license users. For FAQs, refer Licensing FAQs.
- For more details, refer Licensing For Administrators.

Known Issues

- Using multiple monitors with different resolutions can result in issues with the PathWave FPGA UI. We recommend restricting to one resolution of monitor. Below are known issues, but there are likely others:
    - Window does not auto adjust when moving between monitors with different resolutions (e.g. 4K to 2K).
    - Title bar buttons do not respond to user interaction when moved from a 4K monitor to a non-4K monitor if text scaling set at 150% or above.

  - Window cuts off sections of the program on 4K monitors with text scaling set at 250% or above.
  - White border is present around maximized window on 4K monitors with text scaling set at 250% or above.
  - Changing display scaling while PathWave FPGA is running is not recommended and may not work correctly.

- Importing VHDL IP into PathWave FPGA has a number of known limitations. It is recommended to create IP-XACT for any VHDL IP that does not meet the following conditions. A violation of the following conditions will produce a "Syntax Error" message when importing VHDL IP:
  - Port data types must be either `std_logic` or `std_logic_vector`.
  - Port ranges can use generics.
  - Port ranges can use standard math operations (+,-,*,/).
  - Port ranges must start or end with 0 (eg. `din : out std_logic_vector(7 downto 0)` is allowed but `din : out std_logic_vector (7 downto 5)` is not).
- Importing Verilog IP into PathWave FPGA has a number of known limitations. It is recommended to create IP-XACT for any Verilog IP that does not meet the following conditions. Note that only module declarations, port and parameter definitions and 'endmodule' are checked. A violation of the following conditions will produce a "Syntax Error" message when importing Verilog IP:
  - Input/output port sizes may only contain constant values. They may not use parameters or expressions, such as "`input [WIDTH-1:0] x`".
  - When input/output port declarations come after the port list (not ANSI-style/Verilog-2001), all port declarations must appear before any other declarations, such as `parameter`, `reg`, or `signal`.
  - Definition of port attributes is not supported, such as "`(* attribute definition *) input portName,`".
  - When the module declaration contains a parameter list, there must be a space between the module name and the '`#`' for the parameter list.
  - Parameters used in a module declaration may not be defined using parenthesis, unless such a parameter is the last item in the parameter list. (eg: `parameter myParam = (6),`)
  - Port definitions in a module declaration may not be conditionally included using `` `ifdef `` / `` `endif `` statements
  - A module name must include one or more port definitions.
  - To import Verilog source files into PathWave FPGA for use within a design, a module declaration format should be made to conform with of one of the following examples:
    ```
    module foo #( parameter myParam1 = 14, myParam2 = 32) ( input wire clk, output reg [31:0] d_out); endmodule
    ```
    or:
    ```
    module foo (clk, d_out); input wire clk; output reg [31:0] d_out; endmodule
    ```
- When Kactus2 is used for creating IP-XACT for a VHDL file, the VHDL entity declaration must end with "`end <entity_name>`" and not "`end entity`."
- When Kactus2 is used for creating IP-XACT for a Verilog file, avoid comments of the form "`// input name;`" or "`// output name;`" in the Verilog source file as these will cause the Verilog parser to not work properly.

- When using PathWave FPGA remotely on a Windows 7 machine, the frames of the main window and any other dialog of the application may lose their special PathWave FPGA appearance to a more Windows-style one.
- No interconnect exists for PC_MEM interfaces. In the M3202A & M3102A projects this shows up as disallowing multiple memory mapped instances of HVI ports. One Memory mapped port or any number of registers may be placed, but not both at the same time.
    - The program will allow you to place the blocks, but at build time an error will be displayed saying that no PC_MEM interconnect exists.
- Literals are restricted to 64 bits in this release. A '1' in the uppermost bit of the 64 bits can be represented with a hexadecimal or binary representation, or a negative decimal.
- UNC paths are not supported for building FPGA bits.
    - A UNC path can be mapped to a windows drive for building, but this is discouraged due to slow FPGA build times on remote file systems.

System Requirements

You must ensure that your system meets the following requirements before installing PathWave FPGA.

- 2 GB free space on your hard disk drive
- 2 GB RAM (more RAM Recommended)
- Administrator privileges
- Operating system that has the most recent updates and Service Packs
- License File (or Authorization Codes, or token if evaluating) or internet access

Recommended Hardware Configurations

Category	Practical Minimums	Recommended
Operating System	Windows 7 SP1, 64-bit	Windows 10, 64-bit
CPU	Single-core	Quad-core and above
Hard disk	10 GB free space	100 GB free space
RAM	4 GB RAM	16 GB RAM and above
Display	1280 x 720	1920 x 1200
Software Security	USB hardware key	Wired LAN, or Wireless LAN
LAN Connection	Not required	Recommended
Test Instrument Interface	Not required	LAN
Touch User Interface	N/A	Not supported

Note, Windows 8 is not supported.

Summary of Software Compatibility with PathWave FPGA

The following table summarizes PathWave FPGA compatibility with various versions of other software applications. However, for the latest vendor information, licensing, and downloads, please contact each vendor directly.

Vendor	Software / Feature	Release Officially Supported	May work, but not supported	Release Explicitly not-supported
Xilinx	Vivado, debugging, compilation of bit images.	Vivado 2017.3		prior to Vivado 2017.3
CMake	CMake to support to enable FPGA bit file verification	3.9 or later		prior to 3.9
Kactus2	To Import HDL with collapsible interfaces using IP-XACT	3.6 or later	3.5 (note, there is a workaround documented when using parameterized HDL)	3.4
Microsoft	Visual Studio C++ to enable FPGA bit file verification	2017	Other versions	

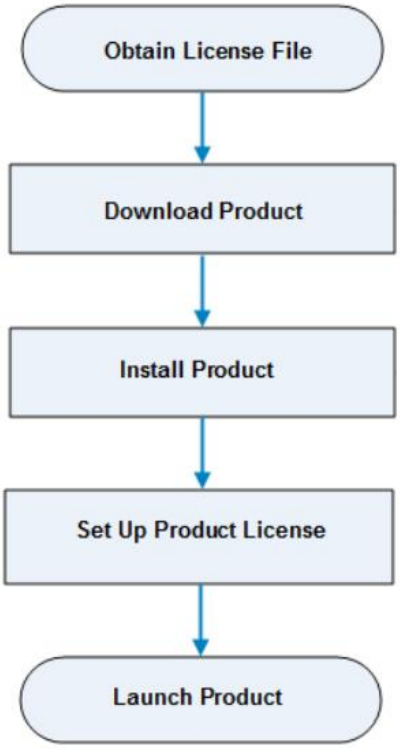
Summary of HDL Language Support

Standard	Release Officially Supported	May work, but not supported	Release Explicitly not-supported
IP-XACT	IEEE 1685-2014		IEEE 1685-2009
Verilog	IEEE 1364-2005		
VHDL	IEEE 1076-2002 (VHDL 2002)		IEEE 1076-2008 (VHDL 2008)

Newer versions of Xilinx Vivado might be required for Keysight Instruments (BSPs). Consult the instrument product manual for specific requirements.

Installation

PathWave FPGA can be installed on a computer running Windows by downloading the PathWave FPGA install file from http://www.keysight.com/find/pathwave_fpga. For the system requirement details, refer System Requirements.



Obtain PathWave FPGA License File

PathWave FPGA requires a license to run. You can either apply for an Evaluation or a Purchased license. Once the license request is approved, a license file (with .lic extension) is sent as an email attachment. Save this file on your computer at *C:\Users\Public*.

Download PathWave FPGA Installer

Click http://www.keysight.com/find/pathwave_fpga to download the installer.

Install PathWave FPGA

To install PathWave FPGA, you must have system administrator privileges. Run the downloaded installer and follow the guided tour to complete the installation. If you want to do a silent install, run the installer executable from the command line as **Administrator** and use the "**--mode unattended**" command line option.

PathWave FPGA License Setup

At the end of installation, the **License Setup Wizard** starts automatically after detecting that you do not have a valid license to start PathWave FPGA. If you choose to skip the license setup, you can complete the process later by clicking **Start > Programs > Keysight PathWave FPGA** <release_number> **> PathWave FPGA** <release_number> **License Manager**.

Node-locked License

To setup a counted license, select the **Add or replace a license file** option and follow the guided tour to complete the license setup process. In case of a USB dongle, attach the dongle to the USB port and invoke the **License Manager** to complete the setup process.

CAUTION	You must have system administrator privileges to setup node-locked licenses (Only) on Windows 7 machines.

Floating License

To setup a floating license, select the **Add or replace a network license server** option and follow the guided tour to complete the license setup process. Consult your license administrator for the network path of the license server.

Launch PathWave FPGA

To run PathWave FPGA, go to the **Start** menu and choose **Programs > Keysight PathWave FPGA** <*release_number*> **> Keysight PathWave FPGA** <*release_number*>.

User's Guide

PathWave FPGA is Keysight's "Open FPGA" development environment. PathWave FPGA provides a complete FPGA design flow from design creation to gateware deployment to HW/gateware verification.

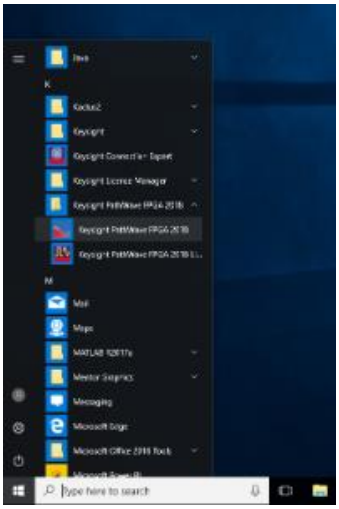
Contents

Overview

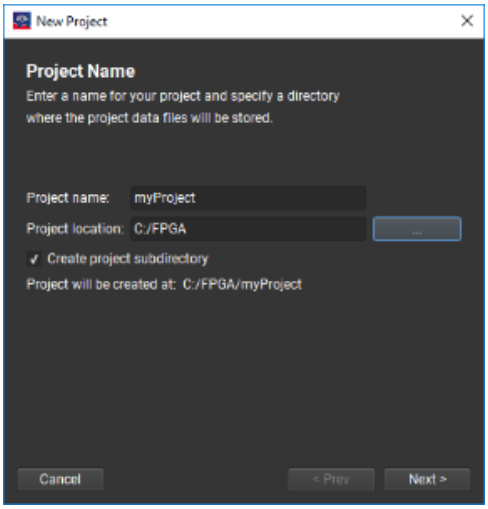
PathWave FPGA is a graphical environment that provides a way to rapidly develop FPGA designs on Keysight Open FPGA hardware. An IP library is provided which includes Logic/Math, Memory, and DSP blocks that can be included in an FPGA design. Vivado IP blocks or custom HDL IP can also be imported and the port interfaces described using IP-XACT 2014. PathWave FPGA provides a design flow from schematic to bitfile generation with the press of a button.

To get started, follow the PathWave FPGA design flow:

1. Start PathWave FPGA

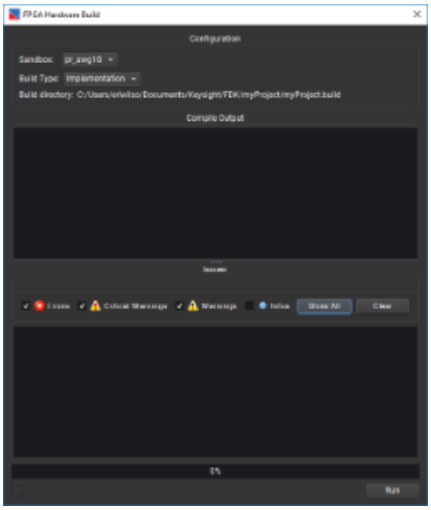


2. Create a new project with the PathWave FPGA New Project Wizard



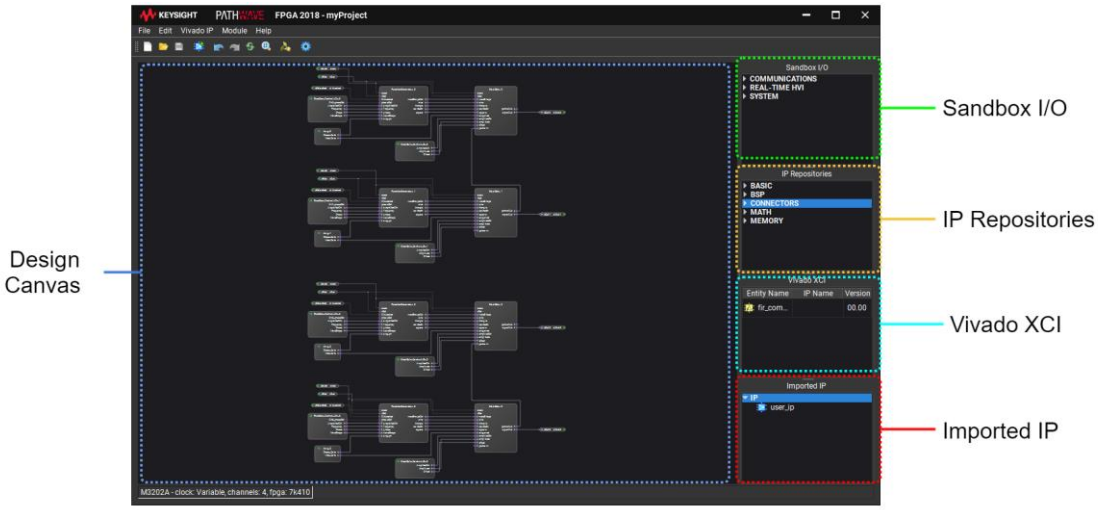
3. Modify the default FPGA template design by importing Vivado IP, HDL IP, or by using the PathWave FPGA IP library.

4. Compile the design into a bit image



5. Deploy your design using the instrument driver or the BSP programming API

GUI Overview



Menu/Icon/Pane	Description
File	Includes options to create a new project, open an existing project, save a project, close a project, add an external block, export to VHDL, create a template, configure settings, and exit.
Edit	Includes options to undo an operation, redo an operation, and select all.
Vivado IP	Includes an option to launch the Vivado IP tool.
Module	Includes and option to generate FPGA firmware.
Help	Includes link to product documentation, license, and product related information.
	Create a new HW project.
	Open an existing project.
	Save the project.
	Add an external IP block.
	Undo the last operation.
	Redo the last operation that was undone.
	Redraw the schematic connections.
	Fit schematic in window.
	Launch the Vivado IP tool.
	Generate the firmware for the project.

Menu/Icon/Pane	Description
Sandbox I/O	Sandbox I/O are responsible for communication between the internally configurable FPGA part (the FPGA customizable space, which a user can edit) and the rest of FPGA.
IP Repositories	IP repositories that are built-in or custom.
Vivado XCI	Vivado XCI (Xilinx Core Instance) created either by launching the Vivado IP tool or importing Vivado XCI. Note, only visible if you have imported a Vivado XCI file.
Imported IP	Imported User IP from many different sources including: VHDL, Verilog, IP-XACT, Vivado Projects (XPR). Note, only visible if you have imported IP.

Keyboard and Mouse Shortcuts

This topic lists the operations that can be performed using keyboard and mouse shortcuts.

- Ctrl + Left click: Zoom in on cursor
- Ctrl + Right click: Zoom out from cursor
- Ctrl + Middle click: Zoom fit to window
- Shift + Left click: Add/remove item from selection
- Shift + Left click and drag: Copy the selection
- Escape: Abort current action
- Delete: Remove selected items
- Ctrl + R: Redraw connections
- Ctrl + F: Zoom fit
- Ctrl + C: Copy selection
- Ctrl + A: Select all
- Ctrl + Z: Undo
- Ctrl + Y: Redo
- Ctrl + N: New project
- Ctrl + O: Open project
- Ctrl + S: Save project
- Ctrl + F4: Close project
- Alt + F4: Exit

Creating a New Project

A hardware project contains the customizable resources of the programmable FPGA of a PathWave FPGA hardware module. When selecting a target module, the project is opened with the factory settings of a standard module. The custom on-board solution is developed within this hardware project and is saved, compiled and loaded into the hardware module (the binary can be loaded into multiple identical modules).

This topic lists the steps to create a new hardware project.

1. Select **File** > **New HW Project**.
2. Enter the project name.
3. Browse to select the project location.

NOTE	To place the project in a subdirectory by the same name, select the **Create project subdirectory** check box.

4. Click **Next**. If a project with the same name exists, a prompt to overwrite the project is displayed. Click **Yes** to overwrite the project.
5. Choose the **Board Support Package** for the target hardware module and click **Next**.
6. Choose a Project Template and click **Next**. A summary of the project details is displayed. Click **Finish**.
7. To save any changes you make to the project, click the **Save** icon or use the menu option.

NOTE	Using the shortcut menu (right-click a block), you can perform the following operations: • To duplicate a block, select **Copy**. • To flip a block horizontally, so inputs are on the right and outputs on the left, select **Flip**. • To redraw the connections to the block, select **Redraw connections**. • To remove the block, select **Remove**. • To view the description/properties, select **Properties**.

Project Directory Structure

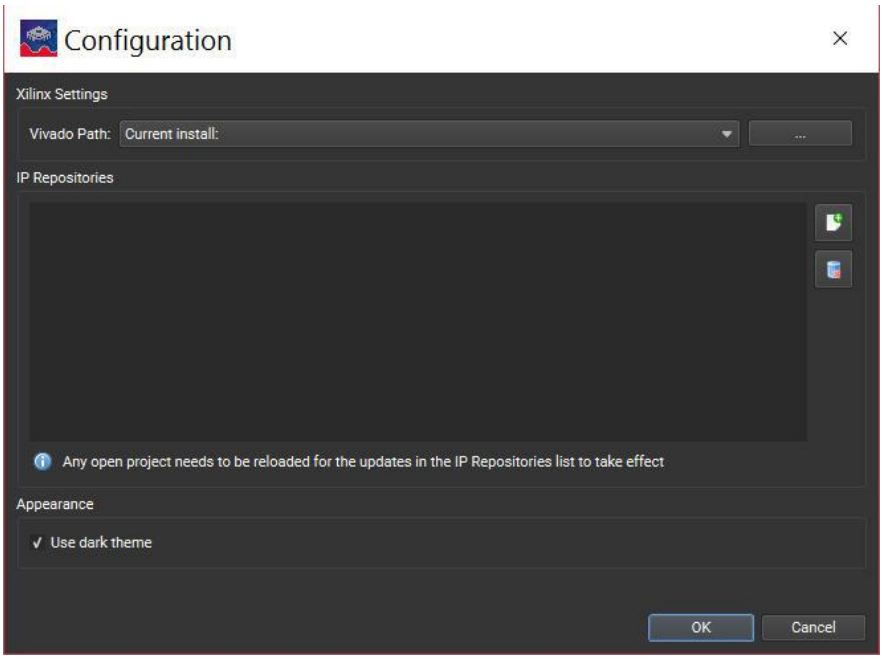
When a new project is created, a project folder with a corresponding project design file is created. This project folder will contain build output and any Vivado XCI (Xilinx Core Instance) IP that you have configured using PathWave FPGA. In the following example, the project created is named *myProject*. The directory structure is shown below:

- **myProject** - Project folder
  - **myProject.kfdk** - Project design file
  - **myProject.build** - Folder containing intermediate build output
  - **myProject.data** - Folder containing final build output and Vivado XCI IP
    - **bin** - Folder with the final build output
      - **myProject_*<timestamp>*** - Folder containing build output
        - **bitgen.log** - Vivado build log file
        - **myProject.k7z** - Program archive that can be downloaded into your FPGA
        - **myProject.spb** - Program FPGA bit file that is an older format, to supported existing instrument software for M3102A, M3202A, M3302A and associated instruments. Newer Keysight hardware will not produce this file output.
    - **VivadoIP** - Folder to contain output for Vivado XCI IP that was configured using PathWave FPGA
      - ***<imported Vivado XCI>*** - Folder for each Vivado XCI IP configured using PathWave FPGA

Configuring PathWave FPGA

The Configuration dialog provides some options for configuring PathWave FPGA. You can specify the Vivado path, IP repositories, and the appearance of the interface.

1. Select **File > Settings**.



- To specify the path to the Vivado installation, browse and select the location. The drop-down box may be used to select between different Vivado versions.
- To add IP repositories, click the **Add Directory** icon. To remove the directories, select the directory and click the **Remove Selected Directories** icon.
- To use the dark theme, select the Use dark theme check box.

IP Repositories

An IP (intellectual property) repository is defined as a library with HDL and the associated IP-XACT describing the HDL. To learn more information on how to create an IP repository, you can review the IP Developers Guide.

> Note, if you change the IP repositories list, you will need to reload the active project.

Limitations

- Currently, PathWave FPGA does not support having multiple IP with the same name. If more that one IP with the same name is encountered during a project load, PathWave FPGA will only load the first one and report an error for the others. To workaround this limitation, you can create a wrapper for your IP with name that does not conflict with any other in the project library.

- IP-XACT 1685-2009 files are not supported. If IP-XACT 1685-2009 files are encountered during the IP repository load process, you will see warning messages.
- When IP repositories loading is completed, you will see an informational message. In case of errors or warnings, the errors will be logged into a temporary file. The temporary file will exist until the closing of PathWave FPGA process. To regenerate the log file, repeat the loading procedure.

Designing Your FPGA Logic

Basic Controls

Zooming In And Out

Using the mouse button

To make the blocks larger: Hold the **Ctrl** key and left-click the mouse on the design canvas as many times as needed to zoom in.

To make the blocks smaller: Hold the **Ctrl** key and right-click the mouse on the design canvas as many times as needed to zoom out.

Using the mouse wheel

To make the blocks larger or smaller: Hold the **Ctrl** key and move the mouse wheel one direction or the other to zoom in or out.

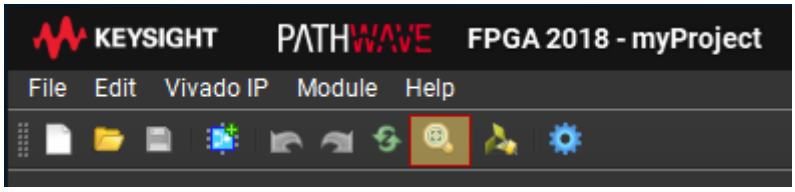
Pan

Hold the **Alt** key and left-click on the mouse and drag to move the project view with the mouse cursor.

Fit in Window

Use the highlighted icon to fit the project within the window if it spills outside the window.

This option is an auto-zoom-out feature to fit all project elements within the window.



Alternatively press **Ctrl** + Middle Click.

Multiple Selections

- To make multiple selections, left-click the mouse and keep the button pressed. Drag a rectangle around the multiple items to be selected.
- Alternatively, hold the **Shift** key and click on the items for multiple selections using the mouse.

Copy Action

To copy a block or element, select the item with the mouse, and use the **Ctrl** + **C** key to copy it. Once the item is copied, the copy can be dragged to the required location.

An alternative way to copy an element is by clicking the **Shift** key, then click on the desired item and move the mouse. Then, the newly copied item can be seen below the mouse cursor, and the item can be dragged and dropped to the required location.

Another way to copy items is to press the right-click mouse button on the item and select the **Copy** option from the shortcut menu.

Move Items

To move an item, left-click the mouse on the item and drag the selected item to the required location.

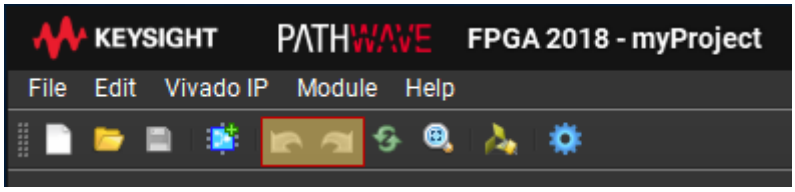
Undo/Redo Action

Using the keyboard:

To **Undo** an action, press the **Ctrl** + **Z** key.

To **Redo** an action, press the **Ctrl** + **Y** key.

Using the GUI toolbar:



Use the Undo or Redo icons.

Using the GUI menu:

Select **Edit** > **Undo** or **Edit > Redo**.

Adding Blocks

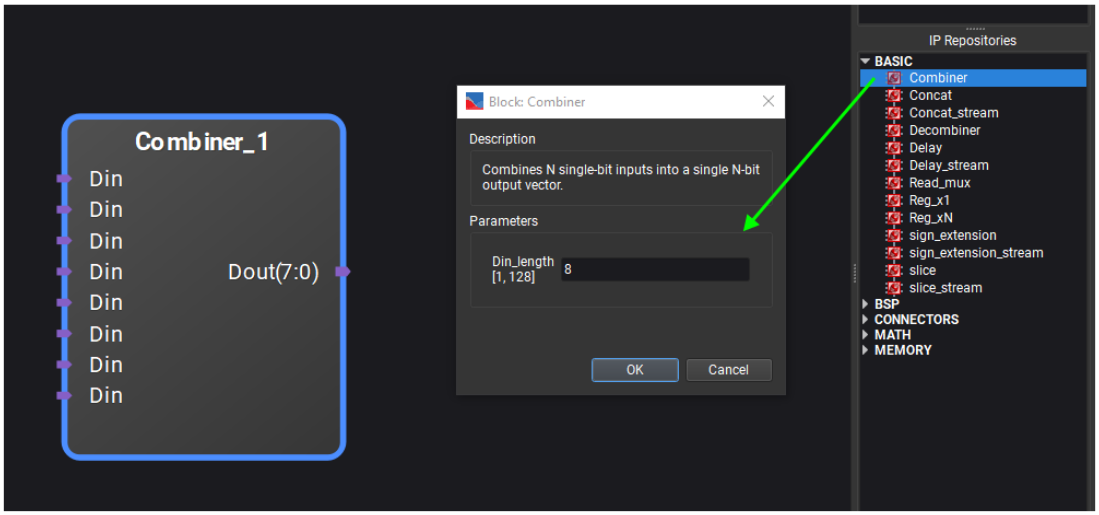
A hardware project is created by combining blocks from the panes displayed on the right side of the user interface. These are grouped under:

- Sandbox I/O

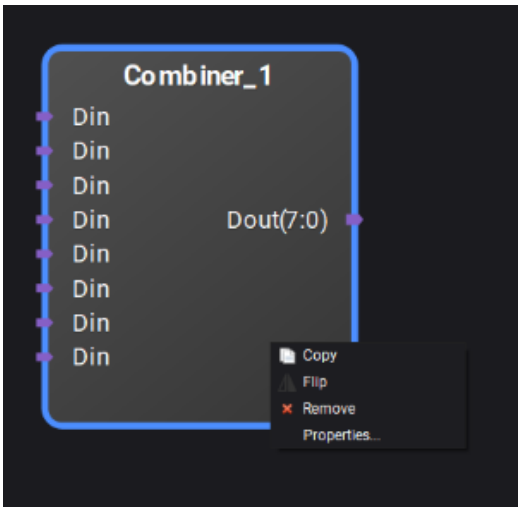
- IP Repositories
- Vivado XCI (Xilinx Core Instance)
- Imported User IP
- PathWave FPGA IP Repository

When a hardware project is opened, sandbox I/O and IP repositories that are available for the particular board support package. The blocks can be selected, dragged into the project, configured, and connected to other blocks in the project.

For example:



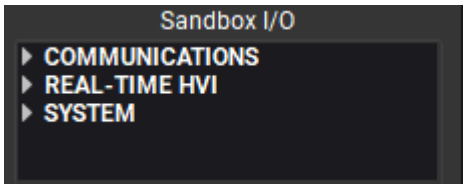
The selected block can be configured and saved.
If you select a block and right-click on it, the following options are available:



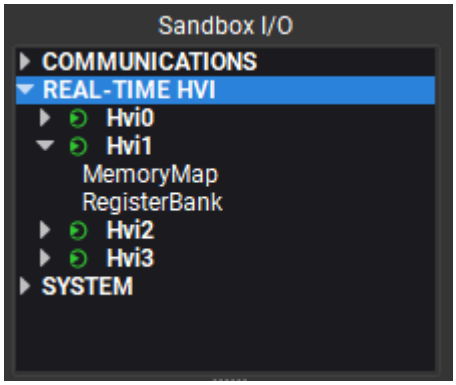
- **Copy** lets you copy this block.
- **Flip** lets you flip the block.
- **Remove** deletes the block from the project.
- **Properties...** provides the configuration dialog box shown above.

Sandbox I/O

To communicate between the sandbox and the static region, you need to instantiate a sandbox I/O block from the Sandbox I/O pane. Each board support package provides a unique set of sandbox I/O blocks that are specific for the instrument. The sandbox I/O blocks are grouped based on the function of their connections to the "outside world". The interfaces of a sandbox are collapsed, in order to show the different categories of sandbox I/O:



Apart from categorizing, some sandbox I/O blocks can instantiated with different types of interfaces. For example, the interface "Hvi1" can be inserted to the schematic as a MemoryMap or connected directly to a RegisterBank.



Finally, it is possible that an interface is comprised only by one port (e.g. a clock). In that case, the interface instance will only show the slot, like in the picture below:

Adding a Register Bank

PathWave FPGA is dedicated to helping customers get their designs ready and tested fast; to facilitate this, PathWave FPGA created Register Banks.

Register Banks are a type of block that can be placed inside the PathWave FPGA schematic. When a register bank is placed in the schematic, PathWave FPGA will generate behind-the-scenes logic to connect the signals that are displayed on the schematic to a memory mapped bus that the customer can access from the Host. By moving this address logic creation inside PathWave FPGA, the user does not have to worry about address overlaps, or decoding blocks. This allows customers to focus their attention on the important parts of their design, and not have to worry about boilerplate components.

How to Create and Update a Register Bank

Below are the steps for creating a Register Bank, and then updating a register bank.

Launching the Register Bank Dialog

1. Launch PathWave FPGA.
2. Open/Create a project you wish to edit.
3. With the project open, in the Sandbox I/O pane, expand **Communications** then expand the interface to which the Register Bank will connect. For the M3102A and M3202A, this will be called **Host**. Under this interface there will be a selection called **RegisterBank**.

4. Either double click on **RegisterBank** or drag **RegisterBank** onto the design canvas to open the Register Bank Dialog.

Creating a Register Bank Using the Register Bank Dialog

With the Register Bank Dialog open you are able to start designing a Register Bank. Register Banks consist of a group of registers with a contiguous address space. Each register in a Register Bank is editable by the user. Below are the major sections of the Register Bank Dialog.

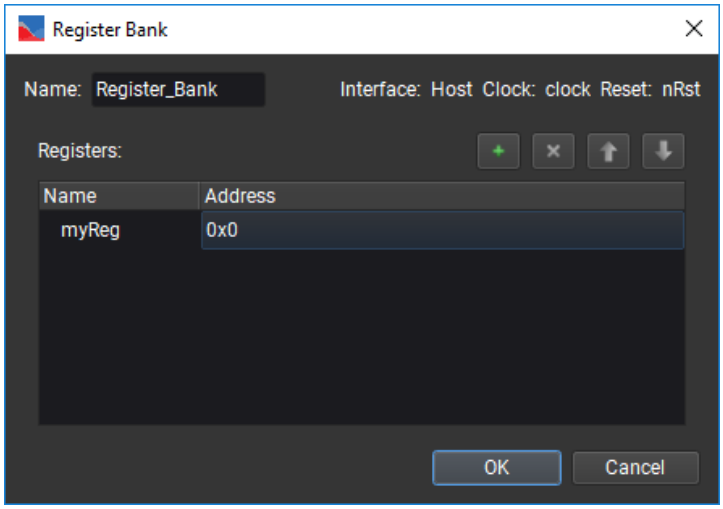


Figure 1: Register Bank Dialog when opened into a new project.

There are 5 main areas to inspect on the Register Bank Dialog

1. Register Bank Name - This is the name that will be displayed on the block when it is placed in the schematic.
    a. The Register Bank Name must be unique, and valid HDL syntax (see Naming Conventions). If the name is not valid, it will be converted to a valid and unique name.
2. Memory Mapped Components - This is the main portion of the Register Bank Dialog. You can edit registers that are contained within the Register Bank here.
    a. Name - This column represents the name of a register. Double left click on the register name to change from the default name. A register name must be unique within the bank, and have valid HDL syntax (see Naming Conventions).
        i. If the Register Dialog detects an issue with the name of a register, it will turn the text red and display a tool tip stating the reason for the failure.
    b. Address - This column represents the byte offset address of a register. The user is not allowed to directly edit this field, it is for informational use only.
    c. Reordering Registers - It is possible to reorder registers in the Register Bank by selecting one, then clicking and dragging it to the location you wish it to go. This changes the address field of the moved register and updates addresses of other registers affected by the move.
3. Add/Remove/Reorder - This section of the dialog is used for manipulating the number and order of registers present in the Register Bank.
    a. The user can add registers to the design if no issues are detected inside the Register Bank. The button will be disabled, when an issue is detected.
    b. The user can remove registers at any point. Any currently selected registers will be removed from the Register Bank.
        i. Another way to remove registers is to use the "Delete" key.
    c. A selected register may be reordered by clicking the up or down arrow.

4. OK/Cancel - This section of the dialog is used to exit the dialog. Clicking OK will create a Register Bank that can be placed on the schematic, while cancel closes the dialog with no other actions taken.
    a. If the dialog detects any issues with the Register Bank, it will disable the "OK" button and display the text "Issue Detected". Please look for the red text to see why the Register Bank is invalid.

Placing the Register Bank in the Schematic

Now that we are done editing the Register Bank, it is time to place the block onto the schematic. To place the block onto the schematic, hit the "OK" button. The block will now be hovered below your cursor. At the location you want to place the block, left click. Below is an example block that was created with default values.

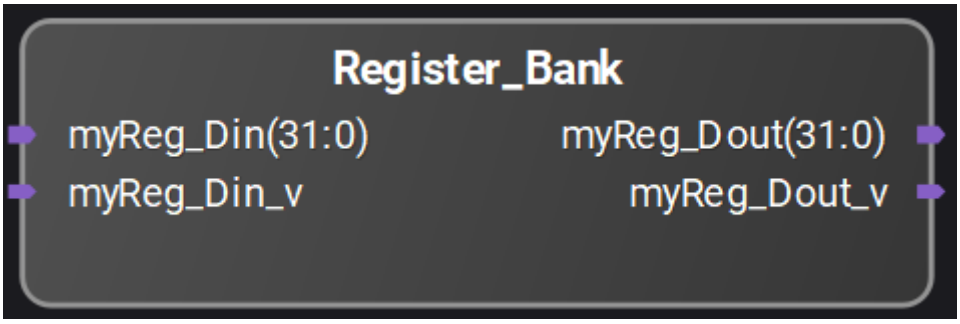


Figure 2: Register Bank block when placed onto the schematic.

Once in the schematic, Register Banks are treated the same as any other block. You are able to move, copy, flip ports, and remove. To use them in your design, just connect the signals displayed on the block to the logic you wish to interact with from the host. PathWave FPGA will handle all of the routing logic for Simulation and Building. You are able to recognize the individual registers in a Register Bank by looking at the names of the signals. The more registers you add to the Register Bank, the more signals will be available. Below is an example of a register block with two registers added to it.

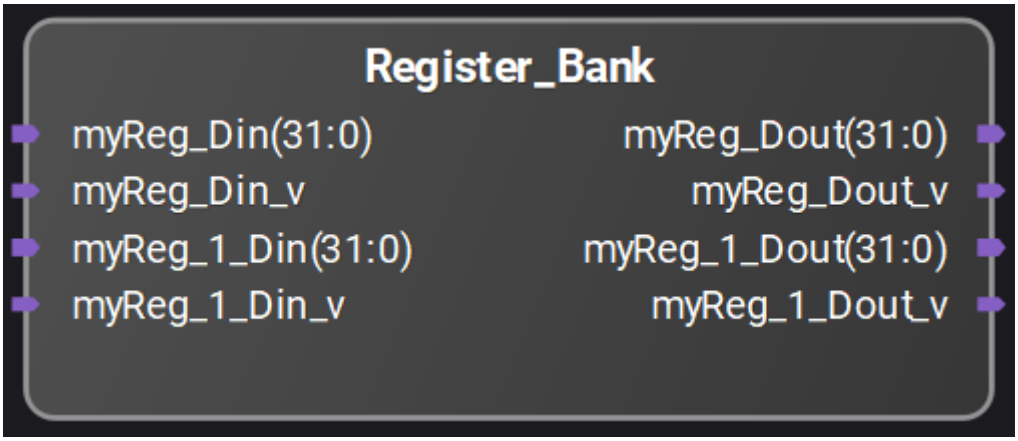


Figure 3: Register Bank block that has two RW registers in it.

Updating Register Banks

A unique feature of Register Banks, is their ability to be modified after they are placed on the schematic. To update the Register Bank we have in Figure 2 to the Register Bank we have in figure 3 we will open the Register Bank Dialog up from the block. There are two ways of opening this dialog.

1. Double click on the Register Bank that you wish to update.
2. Right click on the Register Bank you wish to update, and select "Properties...".

The Register Bank Dialog will open up and display the information that describes the Register Bank you will update.

To add in the second register to our Register Bank, click "Add", then click "OK". Your Register Bank will now have the signals associated with the second register.

If you wish to return your register to the state it was in before the update, simply click the "Undo" Icon in the Icon bar, or use "Ctrl + z".

IP Repositories

IP repositories are libraries of blocks that are loaded into PathWave FPGA. There are three types of IP repositories supported inside PathWave FPGA:

- Default PathWave FPGA IP repository: a repository that is shipped inside the PathWave FPGA Installation directory structure and is permanent. IPs defined in this repository will be loaded for all projects, as long as they meet the hardware support criteria.
- BSP IP repository: a IP repository that is shipped inside a BSP installation.
- User defined IP repository: a user-defined list of directories that include IP definitions. These directories can be defined in the Settings dialog (File → Settings). Important: A project should be reloaded, in order for the added IP to be loaded. To load an IP repository, use the Settings Dialog. To learn how to create an IP repository, refer to the IP Developers Guide.

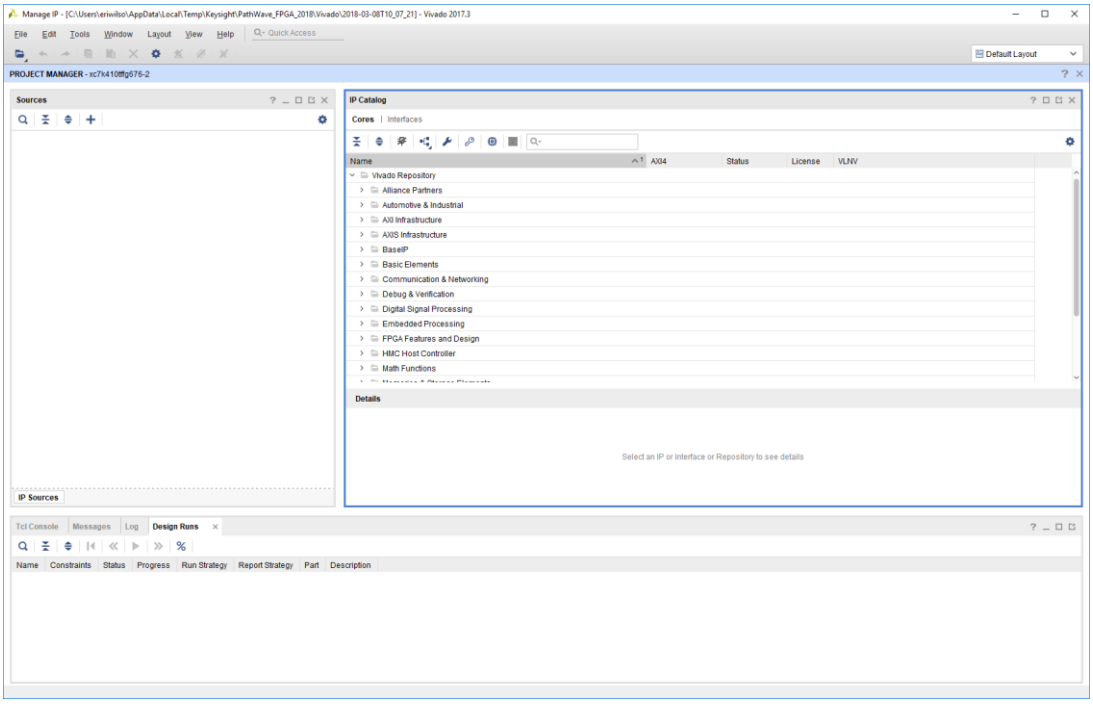
Vivado XCI (Xilinx Core Instance)

Invoking Vivado IP tool

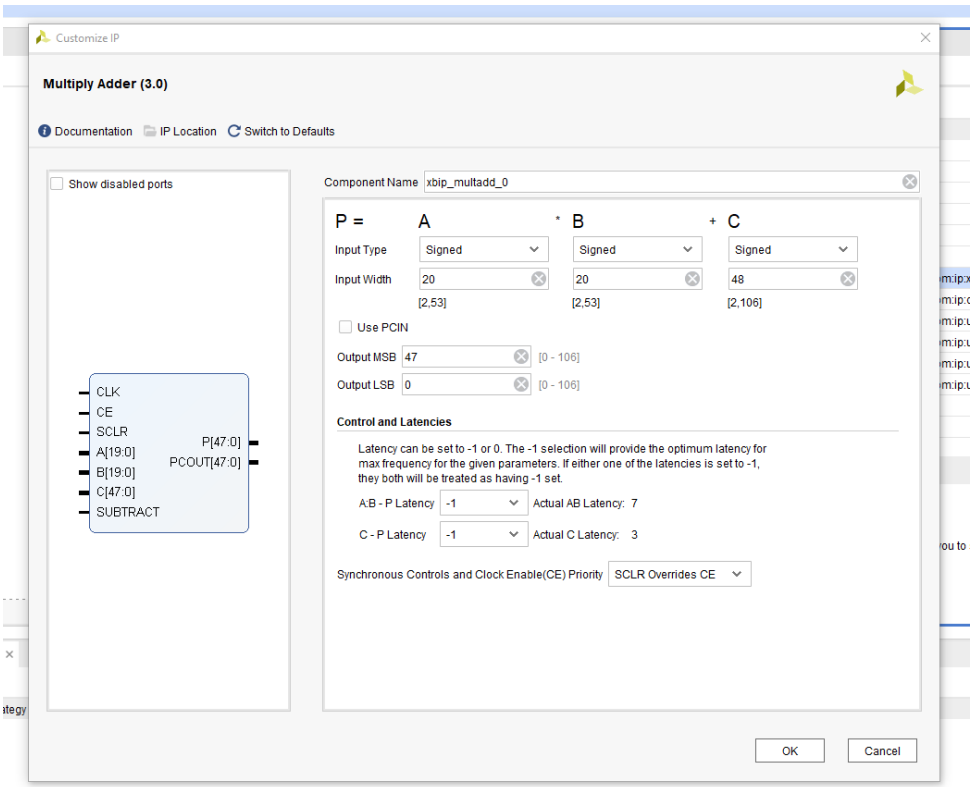
The PathWave FPGA software allows the user to import Vivado IPs from the Xilinx Vivado IP Catalog. The available Vivado IPs can be imported from the catalog and integrated into the project.

To import a Vivado IP:

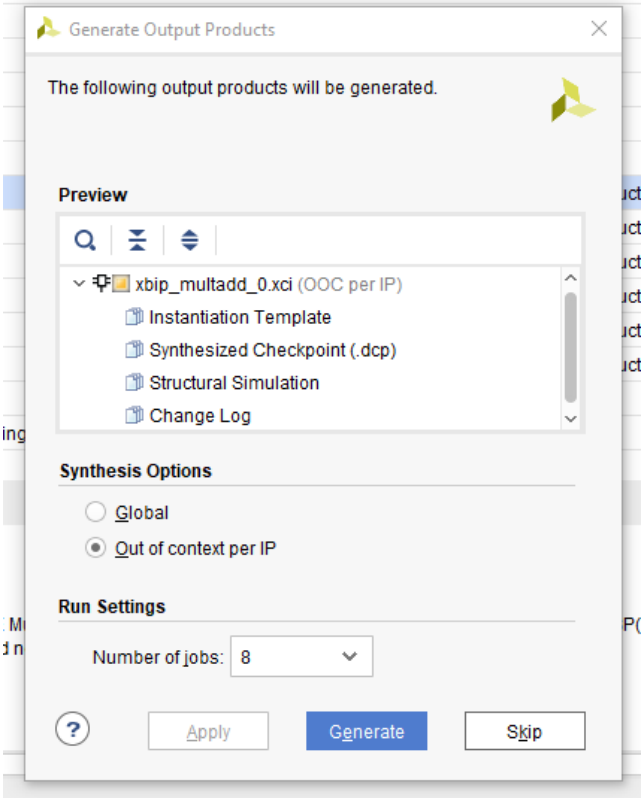
1. Open the PathWave FPGA software.
2. Create a new PathWave FPGA project or open an existing project.
3. Click on the **Launch Vivado IP Tool** icon  .
4. Select a Vivado IP block from the IP Catalog.



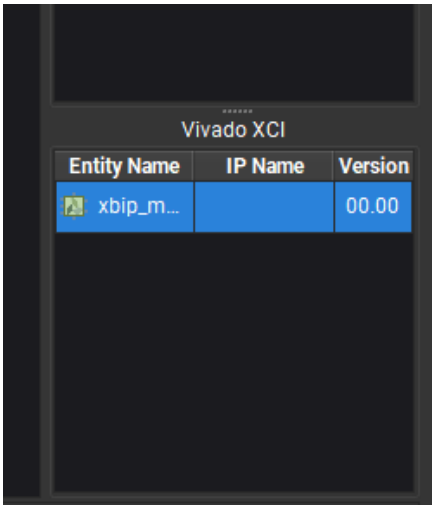
5. Configure the IP properties and then press ok to get to the Generate Output Products dialog window.



6. Click the skip or generate button on the Generate Output Products dialog. Either option will allow for integration into your project.



7. When finished generating or editing Vivado IP close Vivado to add/update the IP in the Vivado IP project.
8. After adding the IP to the project, Vivado IP appears at the bottom-right of the project window:

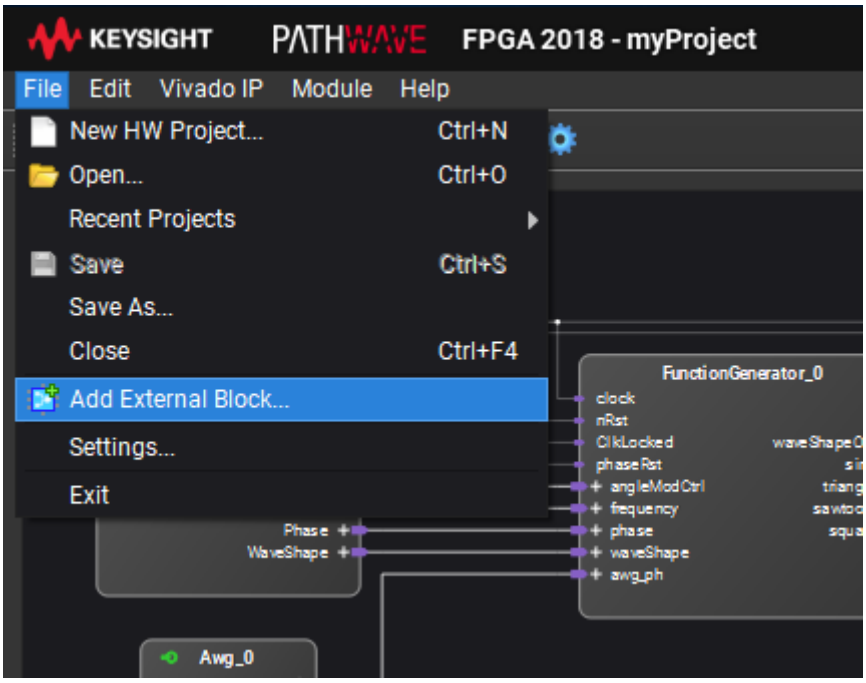


9. Select the Vivado IP and it appears in the project as shown.

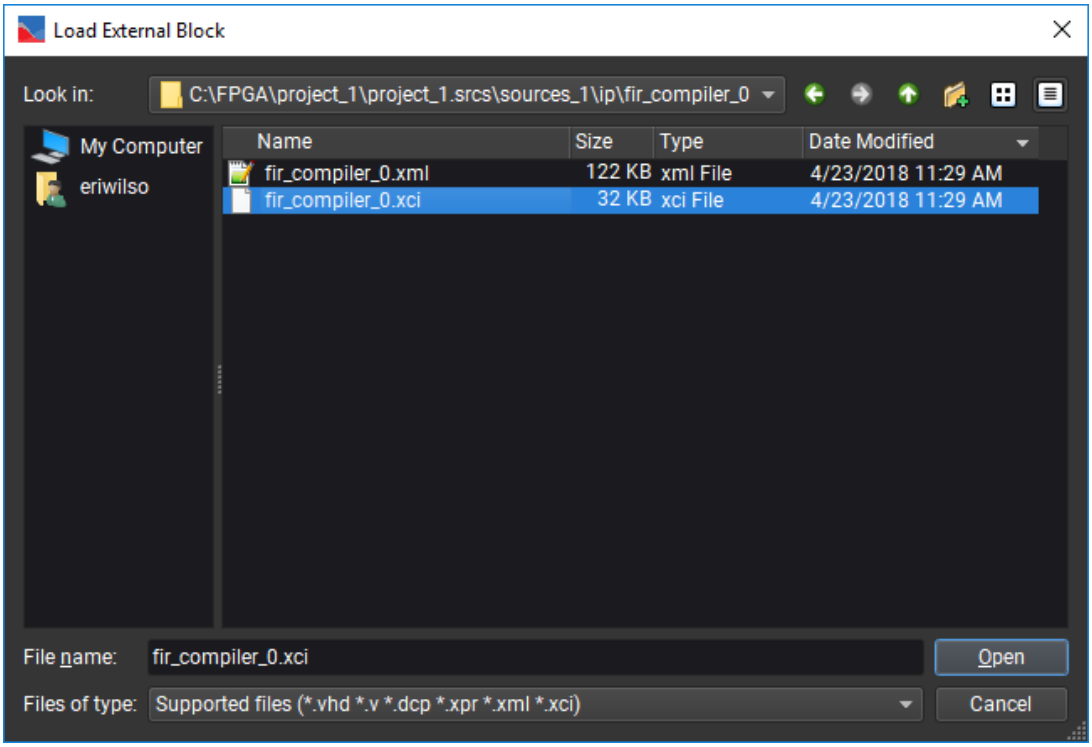
Importing a Vivado XCI File

Another way to import an existing Vivado IP block is to use the Add External Block menu option.

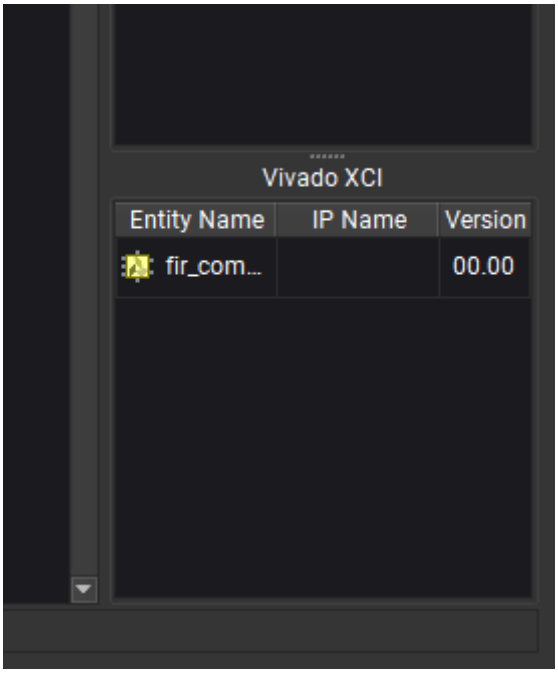
1. Click on the Add External Block menu option



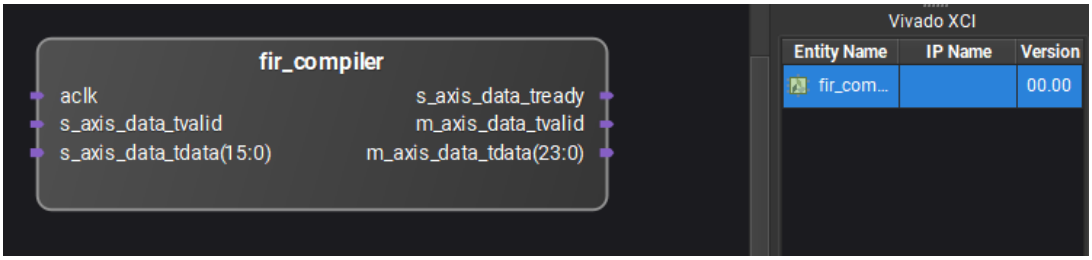
2. Once the Add External Block dialog window opens, navigate to the xci file for the Vivado IP



3. Click Open and the Vivado IP block shows up in the bottom right of the project window.



4. The Vivado IP block can now be used in a design.



Imported User IP

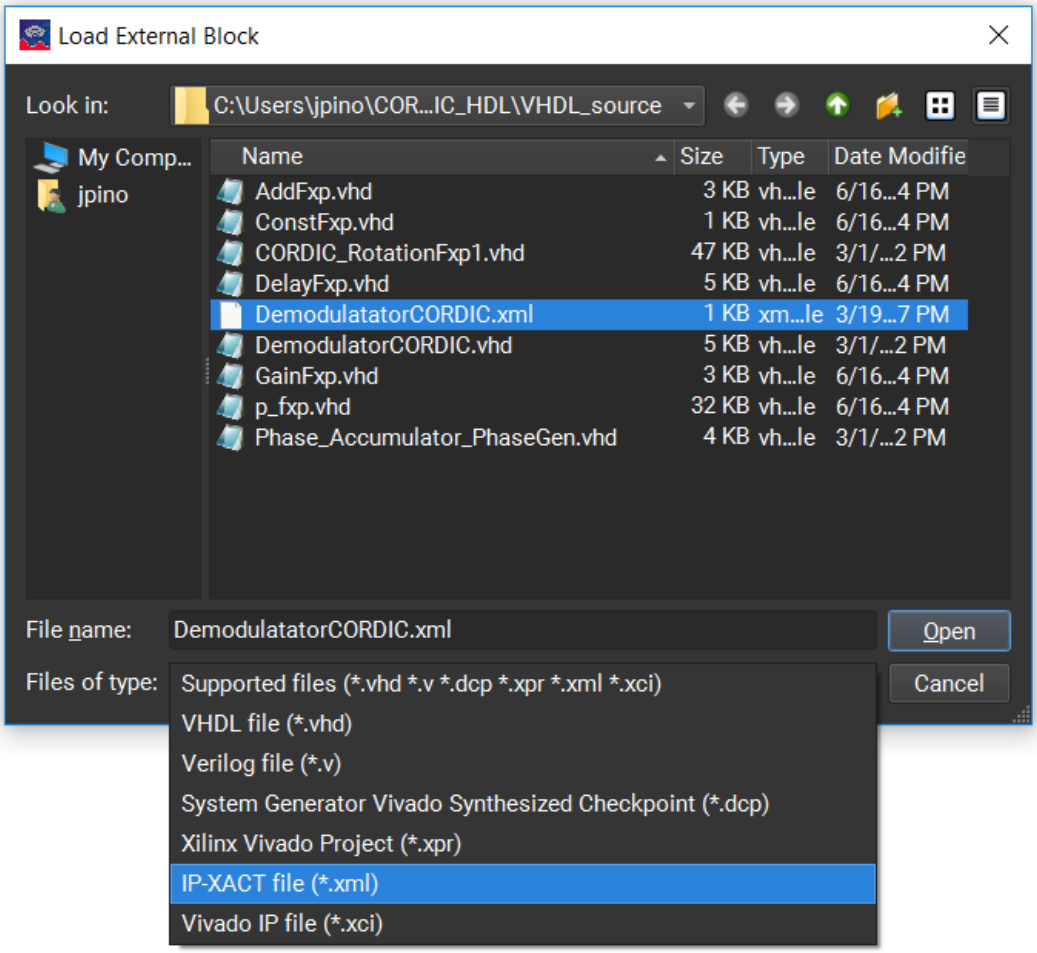
In addition to IPs developed using the Library tools, the PathWave FPGA software allows for importing and integration of user custom IPs into a project. These different user IPs have to be developed by the user, using external FPGA tools. The PathWave FPGA software is not designed for developing IPs from scratch. However, once the user has finished creating an IP (synthesis and simulate it for example), the IP is ready for being imported to the PathWave FPGA software.

The user can import IPs from different source files:

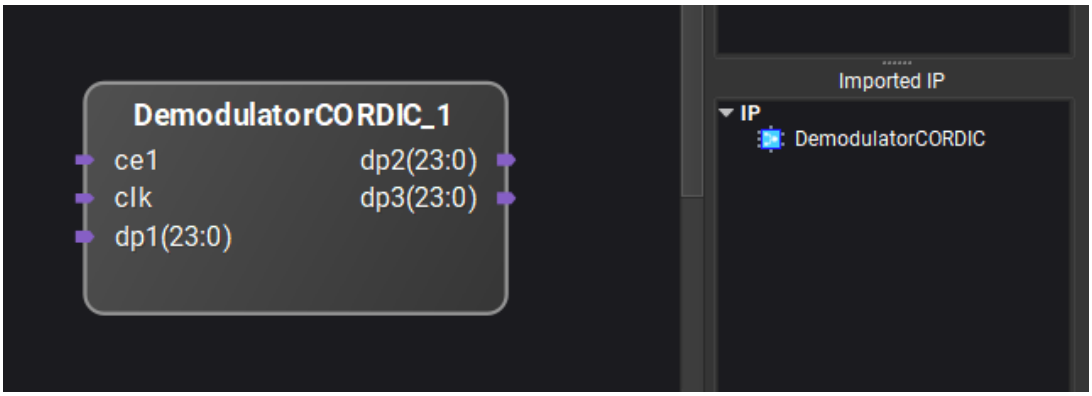
- VHDL source files (*.vhd).
- Verilog source files (*.v).
- Xilinx Vivado projects (*.xpr).
- System Generator Vivado Synthesized Checkpoints (*.dcp).
- IP-XACT files (*.xml).
- Vivado IP files (*.xci)

To import a user IP:

1. Click the icon, or select **File > Add External Block**. In the image below, notice the file types that are available for importing.



2. Select the User IP icon, navigate to select the file to be imported into the project. Click **Open** to import the file.
   The IP is inserted in the project, where it can be connected to other blocks.



The block name appears in the User IP External Block region for reuse as shown above. To remove a block, right-click the block name and choose remove.

- If the User IP file is moved, an "**X**" appears at the top of the block indicating the file cannot be found. Once the file is moved back, or the path is changed, right-click the block to reload the IP and remove the "**X**" on the block.

- If the underlying code for the IP is changed, a "**!**" can appear to signify an alert condition. Once the code is corrected, the block can be reloaded to remove the "**!**" on the block.

Importing an HDL file with Dependencies

If you want to import an HDL file with dependencies, you will need to create an IP-XACT file for the desired HDL entity following the instructions in the IP Developers Guide. Then, inside the `<ipxact:fileset>` where the source files for "synthesis" are supposed to be defined, the user has to add as many `<ipxact:file>` entries as are required to define the source VHDL file along with all the files that it is depending on.

For example, let's assume that the desired component is called "Filter" and is defined in "`C:\MyIPs\FilterIP\FilterTop.vhd`". Then, let's say the implementation of "Filter" depends on another component, named "Tap", which is defined in "`C:\MyIPs\FilterIP\Tap.vhd`". To successfully load the component "Filter" in PathWave FPGA, we need to create an IP-XACT (e.g. in "`C:\MyIPs\FilterIP\Filter.xml`") file with the following statements in the fileset entry:

Code Block 1 IP-XACT fileset snippet

```
<ipxact:fileSets>
   <ipxact:fileSet>
      <ipxact:name>synthesis</ipxact:name>
      <ipxact:file>
         <ipxact:name>FilterTop.vhd</ipxact:name>
         <ipxact:fileType>vhdlSource</ipxact:fileType>
      </ipxact:file>
      <ipxact:file>
         <ipxact:name>Tap.vhd</ipxact:name>
         <ipxact:fileType>vhdlSource</ipxact:fileType>
      </ipxact:file>
   </ipxact:fileSet>
</ipxact:fileSets>
```

When the IP-XACT file is created, you can use the process above to load the IP-XACT xml file.

Importing a HDL file without Dependencies

When an HDL file is imported without dependencies, only the module or entity declaration will be examined in order to determine the ports that will be available for connections within a PathWave FPGA graphical design. Any syntax issues or errors that may exist elsewhere in an imported HDL file may not be detected or flagged.

For Verilog HDL files, module declarations should be limited to the features and format shown in the following examples:

```
module foo (clk, d_out);
input wire clk;
output reg [31:0] d_out;

endmodule
```

or:

```
module foo
#(
    parameter myParam1 = 14,
    parameter myParam2 = 32
)
(
    input wire clk,
    output reg [31:0] d_out
);

endmodule
```

or:

```
module mymodule(input         clk,
                input [7:0]   inBus,  // Comments are okay
                output        outWire,
                output [15:0] outBus);

endmodule
```

For VHDL source files, entity declarations should be limited to features shown in the following example:

```
library ieee;
use ieee.std_logic_1164.all;

entity foo is
    generic (
        width   : integer := 4
    );
 port (
    clk : in  std_logic;
    d_out: out std_logic_vector(width-1 downto 0)
 );
end foo;
```

A list of known Verilog and VHDL limitations for IP import can be found in the Release Notes.

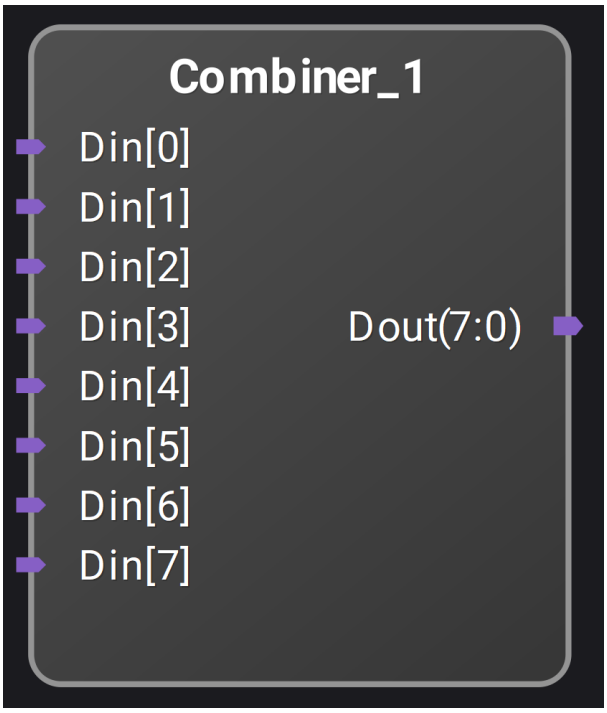
PathWave FPGA IP Repository

PathWave FPGA includes some IP blocks that a user can incorporate into their FPGA design. The IP blocks are categorized into different libraries so that similar blocks are grouped together. Below is a description of the IP blocks included in PathWave FPGA.

Some of the IP blocks are designed so that they can optionally process multiple samples in the same clock. This is called *supersampling*. For blocks that support this, there is a parameter called *supersample* that denotes the number of parallel samples. For example, a 32 bit adder with supersample=1 would add two 32 bit numbers. A 32 bit adder with supersample=2 would add two pairs of 16 bit numbers. This can be useful when processing data at a higher sample rate than the clock rate of the FPGA.

Basic IP blocks

Combiner

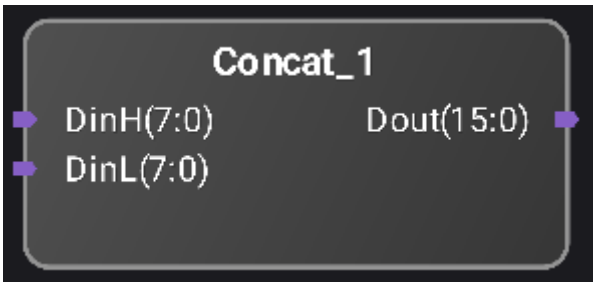


Combines N single-bit inputs into a single N-bit output vector.

Parameters

Din width: Sets the number of single bit inputs. Variable from 1 to 128. Default is 8.

Concat



Concatenates two input signals into one single signal. DinH is the most significant half of Dout, and DinL is the least significant half of Dout.

This module does not introduce extra delay.

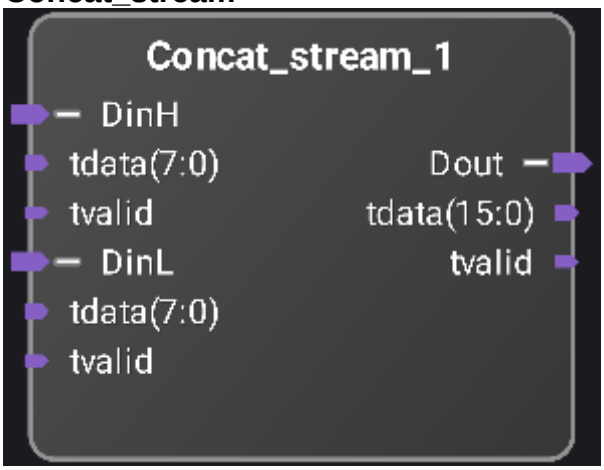
Parameters

DinH width: Sets the data width of DinH. Variable from 1 to 1024. Default is 8.

DinL width: Sets the data width of DinL. Variable from 1 to 1024. Default is 8.

supersample: Sets the supersample amount. Variable from 1 to 64. Default is 1.

Concat_stream



Streaming version of the concat block.

Concatenates two input signals into one single signal. DinH is the most significant half of Dout, and DinL is the least significant half of Dout

This module does not introduce extra delay.

Note that both streaming inputs must assert and deassert tvalid at the same time.

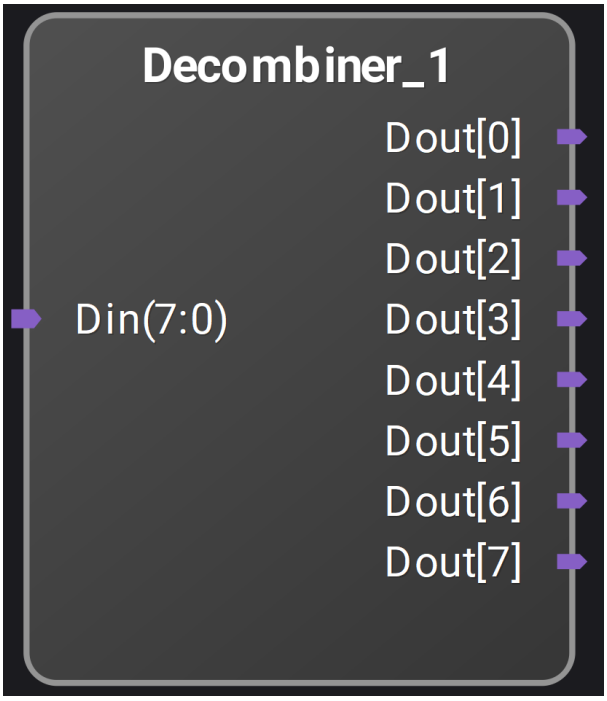
Parameters

DinH width: Sets the data width of DinH. Variable from 1 to 1024. Default is 8.

DinL width: Sets the data width of DinL. Variable from 1 to 1024. Default is 8.

supersample: Sets the supersample amount. Variable from 1 to 64. Default is 1.

Decombiner

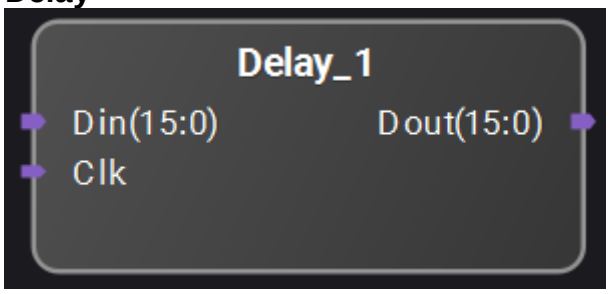


Converts a single N-bit input vector into N single-bit output signals.

Parameters

Din width: Sets the Din data width. Variable from 1 to 128. Default is 8.

Delay



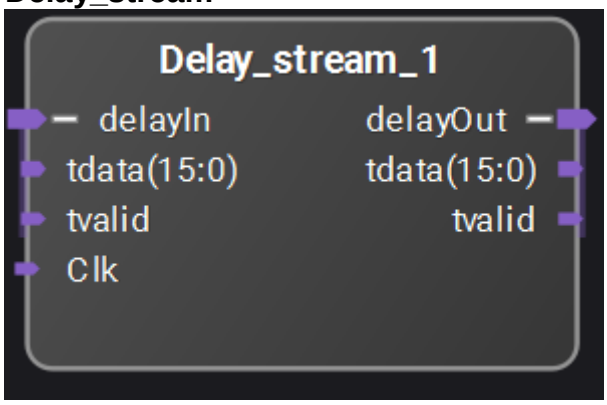
Delays input N cycles.

Parameters

bus width: Sets the bus width of Din and Dout. Variable from 1 to 1024. Default is 16.

latency: Sets the latency through the delay block. Variable from 1 to 1024. Default is 1.

Delay_stream



Streaming version of the delay block.

Delays input N cycles.

Parameters

bus width: Sets the bus width of Din and Dout. Variable from 1 to 1024. Default is 16.

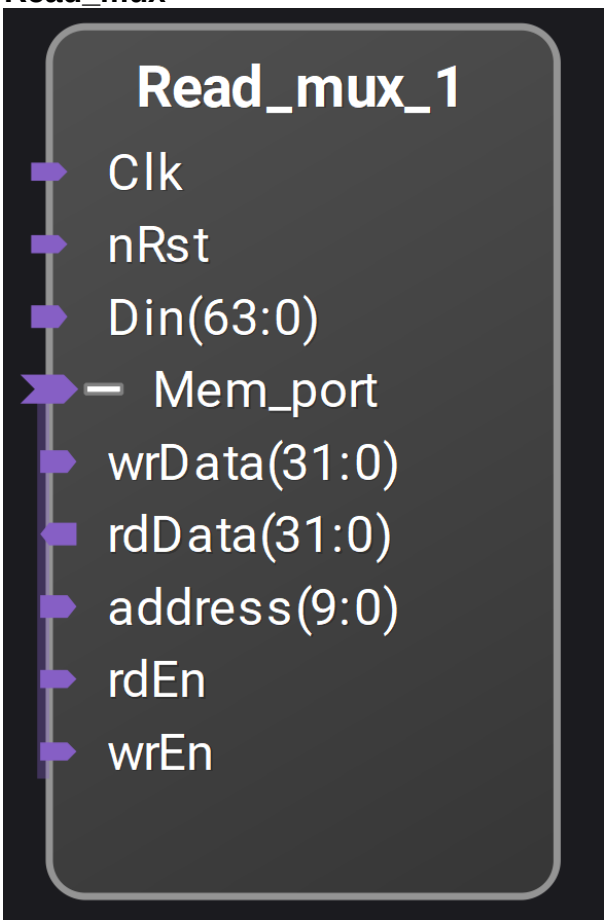
latency: Sets the latency through the delay block. Variable from 1 to 1024. Default is 1.

Latch

32 bit latch with write enable.

Parameters
Bus width: Sets the register bus width. Variable from 1 to 1024. Default is 32.

Read_mux

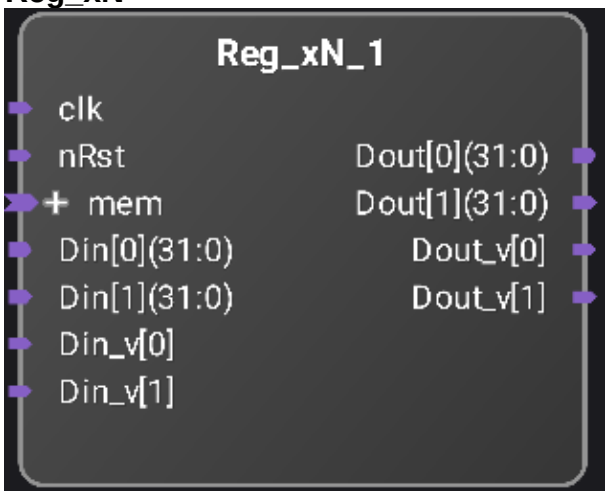


Read data from multiple sources.

Address port is used to select one of N, 32 bit data sources. If the address index is larger than the number of input data sources, this block will return zeros.

Parameters
Number of inputs: Sets the bus width of Din in 32 bit increments. Default is 2.

Reg_xN



Captures N, 32 bit data inputs and drives to outputs. The internal data register may be updated through a write access on the 'mem' port indexed by the address value. The internal data register may also be updated to the Din value by asserting the corresponding Din_v signal[n]. When both updates are attempted at the same time, the mem write value will take precedence. The values of the internal data registers are driven out the Dout[n] ports.

Mem read access will return the value of the indexed internal data register.

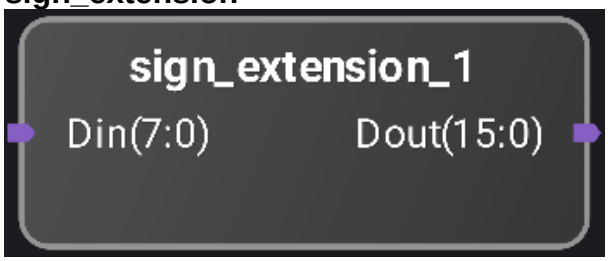
The Dout_v[n] signal is asserted high for one clock period when new data is written. This is any time a mem write occurs or when Din_v[n] is asserted.

Parameters

Number of Registers: Variable from 1 to 256. Default is 2.

Address width: Variable from 1 to 1024. Default is 32.

sign_extension



Sign extends the input vector.

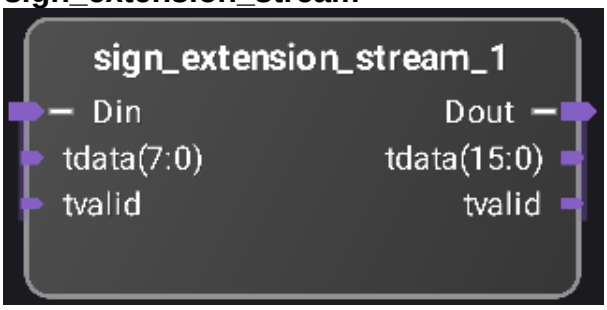
Parameters

Din width: Sets the Din bus width. Variable from 1 to 1024. Default is 8.

Dout width: Sets the Dout bus width. Variable from 1 to 1024. Default is 16.

supersample: Sets the supersample amount. Variable from 1 to 64. Default is 1.

sign_extension_stream



Sign extends the input vector.

Parameters

Din width: Sets the Din bus width. Variable from 1 to 1024. Default is 8.

Dout width: Sets the Dout bus width. Variable from 1 to 1024. Default is 16.

supersample: Sets the supersample amount. Variable from 1 to 64. Default is 1.

slice

Selects certain number of bits from a vector signal input.

This does not introduce extra delay.

Parameters

Din width: Sets the bus width of Din. Variable from 1 to 1024. Default is 16.

offset: Sets the starting LSB position to extract bits from Din [Dout(bus_out_width:0) = Din(bus_in_width:offset_lower_bit)]. Default is 0.

Dout width: Sets the bus width of Dout. Variable from 1 to 1024. Default is 16.

supersample: Sets the supersample amount. Variable from 1 to 64. Default is 1.

slice_stream

Streaming version of the slice block.

Selects certain number of bits from a vector signal input.

This does not introduce extra delay.

Parameters

Din width: Sets the bus width of Din. Variable from 1 to 1024. Default is 16.

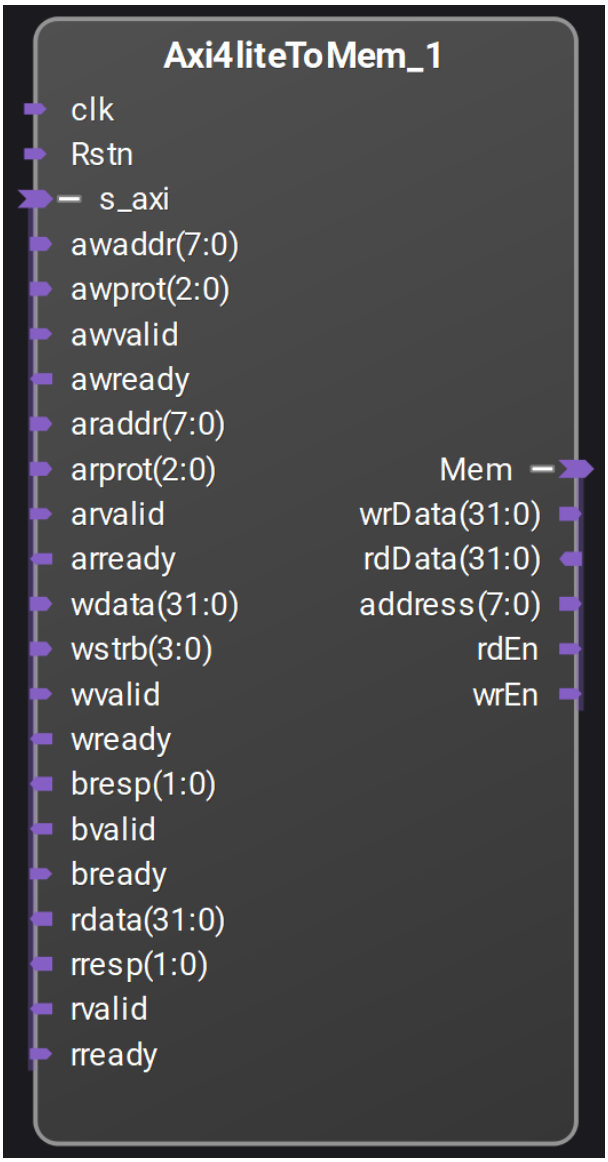
offset: Sets the starting LSB position to extract bits from Din [Dout(bus_out_width:0) = Din(bus_in_width:offset_lower_bit)]. Default is 0.

Dout width: Sets the bus width of Dout. Variable from 1 to 1024. Default is 16.

supersample: Sets the supersample amount. Variable from 1 to 64. Default is 1.

Connectors

Axi4liteToMem



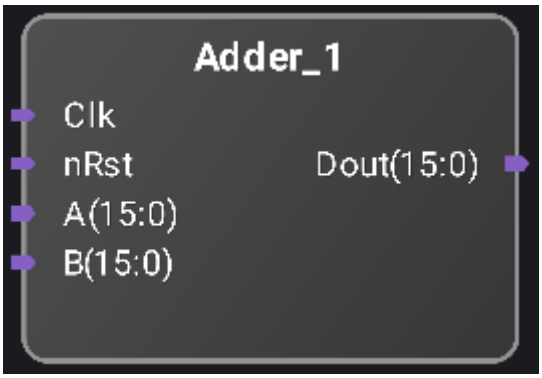
Converts Axi4Lite slave interface to PC Mem master interface.

Parameters

Address width: Sets the AXI interface and Mem interface address width. Default is 8.

Math

Adder



Signed adder.

Inputs are expected to have the same length.

Overflow and underflow check is done when saturate is enabled.

Output width is increased by 1 when full precision is enabled.

Subtraction changes operation from A+B to A-B.

This module adds a delay of 1 cycle.

When latch input is enabled, 1 extra cycle of delay is added.

Parameters

input width: Sets the bus width of the A and B inputs. Default is 16.

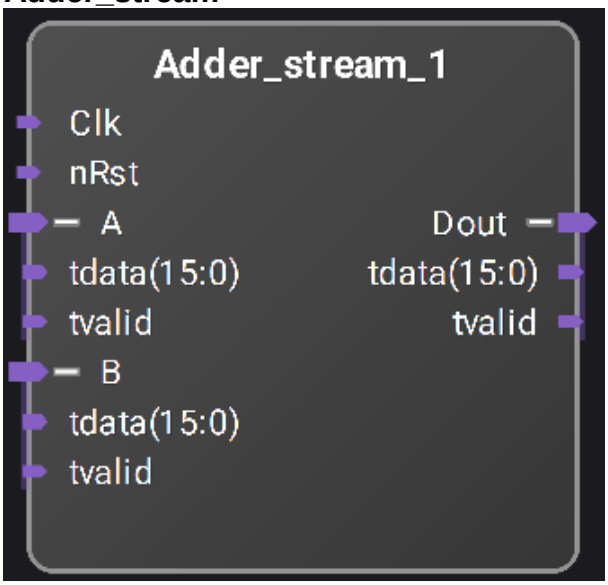
Adder implementation: Selects saturate or full precision adder modes. Default is Saturate.

latch input: When enabled the data on the A and B inputs is latched. Default is no latch.

subtract: When enabled the adder operation is changed from A+B to A-B. Default is add.

supersample: Sets the supersample amount. Variable from 1 to 64. Default is 1.

Adder_stream



Signed adder.

Inputs are expected to have the same length.

Overflow and underflow check is done when saturate is enabled.

Output width is increased by 1 when full precision is enabled.

Subtraction changes operation from A+B to A-B.

This module adds a delay of 1 cycle.

When latch input is enabled, 1 extra cycle of delay is added.

Parameters

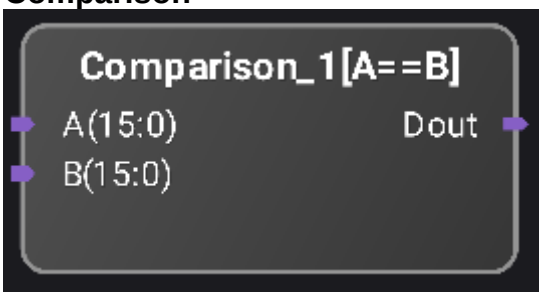
input width: Sets the bus width of the A and B inputs. Default is 16.

Adder implementation: Selects saturate or full precision adder modes. Default is Saturate.

subtract: When enabled the adder operation is changed from A+B to A-B. Default is add.

supersample: Sets the supersample amount. Variable from 1 to 64. Default is 1.

Comparison



Comparisons between inputs A and B.

Output is set to one when the comparison set by the operation parameter is true.

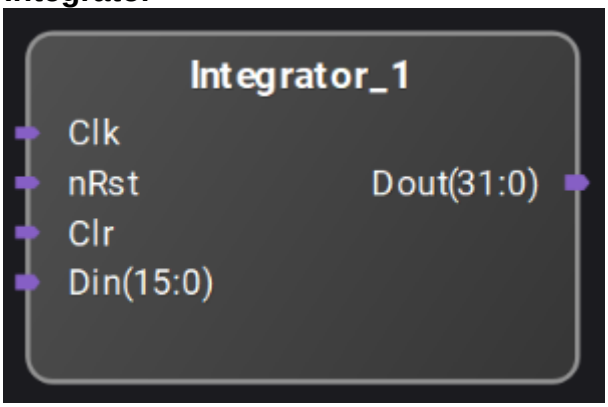
Parameters

operation: Select between A==B, A!=B, A>B, A<B, A>=B, and A<=B. Default is A==B.

data size: Sets the bus width of the A and B inputs. Default is 16.

sign: Select when the data on the A and B inputs is signed. Default is unsigned.

Integrator



Data integrator.

When selecting signed input, sign extension is automatically applied.

The internal accumulator can be reset by the nRst or Clr inputs.

When supersample > 1, all the input samples are summed into the same internal accumulator.

This module adds a delay of 1 cycle by default.

When latch input is enabled, an extra cycle of delay is added.

Parameters

input_width: Sets the bus width of the input samples. Variable from 1 to 1024. Default is 16.

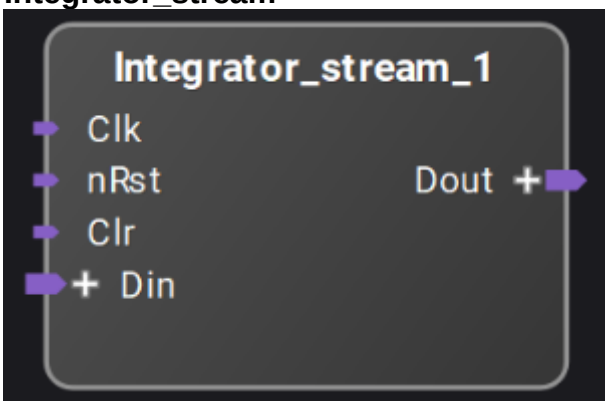
output_width: Sets the bus width of the internal accumulator and the output. Variable from 1 to 1024. Default is 32.

input_signed: When enabled, the input samples represent signed values and will be sign extended prior to accumulation. Default is unsigned.

latch input: When enabled, the input data is latched prior to accumulation. This adds a cycle of delay. Default is no latch.

supersample: Sets the supersample amount. All the input samples are summed into the same internal accumulator. Variable from 1 to 64. Default is 1.

Integrator_stream



Data integrator with streaming interface.

When selecting signed input, sign extension is automatically applied.

The input samples are accumulated oly when the tvalid signal is asserted.

The internal accumulator can be reset by the nRst or Clr inputs.

When supersample > 1, all the input samples are summed into the same internal accumulator.

This module adds a delay of 1 cycle by default.

When latch input is enabled, an extra cycle of delay is added.

Parameters

input_width: Sets the bus width of the input samples. Variable from 1 to 1024. Default is 16.

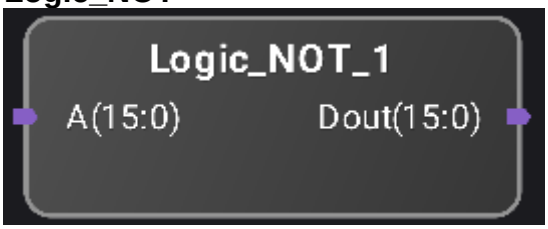
output_width: Sets the bus width of the internal accumulator and the output. Variable from 1 to 1024. Default is 32.

input_signed: When enabled, the input samples represent signed values and will be sign extended prior to accumulation. Default is unsigned.

latch input: When enabled, the input data is latched prior to accumulation. This adds a cycle of delay. Default is no latch.

supersample: Sets the supersample amount. All the input samples are summed into the same internal accumulator. Variable from 1 to 64. Default is 1.

Logic_NOT

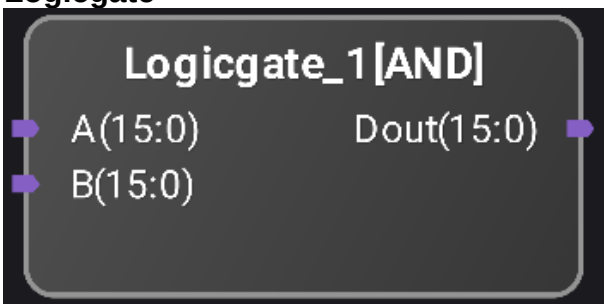


Logic NOT operation.

Parameters

data size: Sets the bus width of the A and Dout ports. Variable from 1 to 1024. Default is 16.

Logicgate



Output is the logical operation between inputs A and B.

The operation parameter determines which logical operation is performed from AND, OR, XOR, NAND, NOR, and XNOR.

Parameters

data size: Sets the bus width of the A, B, and Dout ports. Variable from 1 to 1024. Default is 16.

operation: Selects one of the logic operations listed above. Default is AND.

Multiplier

Multiplier (DSP core).

Input lengths and signedness are configurable.

When both inputs are signed, output length is the sum of both inputs lengths minus 1.

This block adds a delay of 1 cycle.

Latch input increases the total delay by an additional clock cycle.

When the Dout width is less than Input A width + Input B width, Dout will consist of the lower bits of the product.

Parameters

A width: Sets the bus width of the A input. Variable between 1 and 1024. Default is 16.

A signed: Select when the A input data is signed. Default is unsigned.

B width: Sets the bus width of the B input. Variable between 1 and 1024. Default is 16.

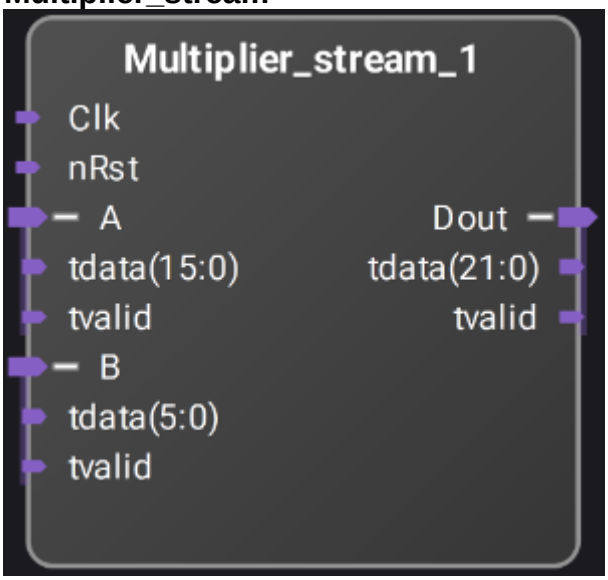
B signed: Select when the B input data is signed. Default is unsigned.

Dout width: Sets the bus width of the Dout port. Variable between 1 and 1024. Default is 16.

Latch input: Input data is latched when selected. Default is no latch.

supersample: Sets the supersample amount. Variable between 1 and 64. Default is 1.

Multiplier_stream



Multiplier (DSP core).

Input lengths and signedness are configurable.

When both inputs are signed, output length is the sum of both inputs lengths minus 1.

This block adds a minimum delay of 1 cycle.

Pipeline increases the total delay by an additional clock cycle.

When the Dout tdata width is less than Input A tdata width + Input B tdata width, Dout will consist of the lower bits of the product.

Parameters

A width: Sets the bus width of the A input. Variable between 1 and 1024. Default is 16.

A signed: Select when the A input data is signed. Default is unsigned.

B width: Sets the bus width of the B input. Variable between 1 and 1024. Default is 16.

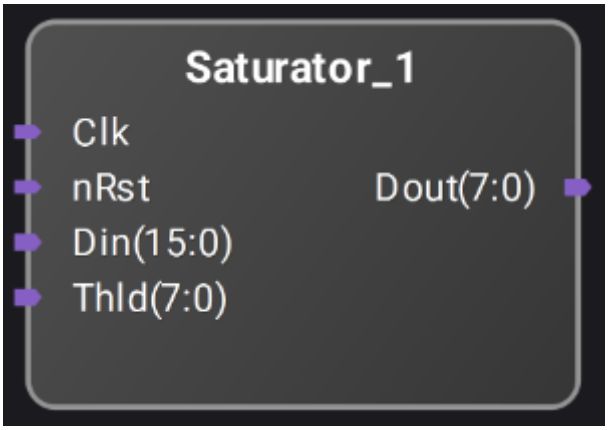
B signed: Select when the B input data is signed. Default is unsigned.

Dout width: Sets the bus width of the Dout port. Variable between 1 and 1024. Default is 16.

pipeline: When selected a pipelined multiplier is used. Default is no pipelining.

supersample: Sets the supersample amount. Variable between 1 and 64. Default is 1.

Saturator



Output data is set to a saturation value (set by Thld port) whenever input data is equal or greater than that value.

For signed data, output data is set to a saturation value (-Thld) whenever input data is less than that value.

Saturation value can not be greater than the maximum possible value of the output vector.

Parameters

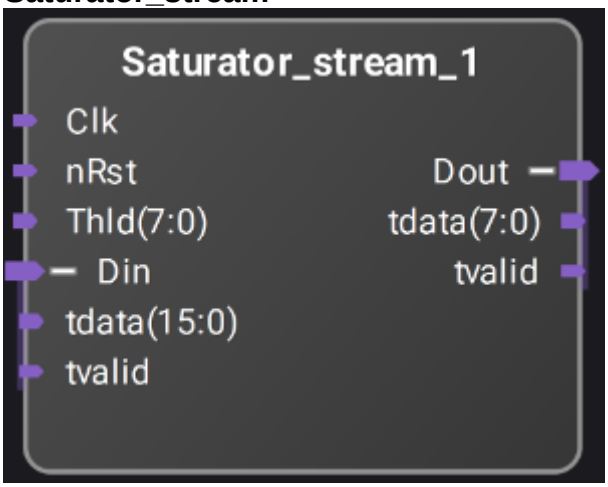
Din signed: Select when the data on the Din input is signed. Default is signed.

Din width: Sets the Din bus width. Variable between 1 and 1024. Default is 16.

Dout width: Sets the Dout bus width. Variable between 1 and 1024. Default is 8.

supersample: Sets the supersample amount. Variable between 1 and 64. Default is 1.

Saturator_stream



Output data is set to a saturation value (set by Thld port) whenever input data is equal or greater than that value.

For signed data, output data is set to a saturation value (-Thld) whenever input data is less than that value.

Saturation value can not be greater than the maximum possible value of the output vector.

Parameters

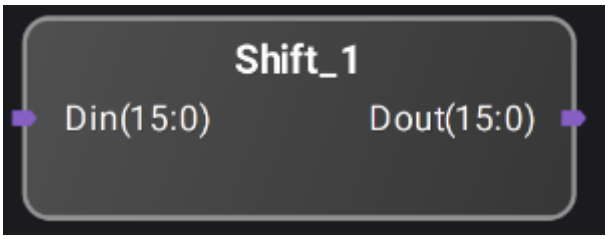
Din signed: Select when the data on the Din input is signed. Default is signed.

Din width: Sets the Din bus width. Variable between 1 and 1024. Default is 16.

Dout width: Sets the Dout bus width. Variable between 1 and 1024. Default is 8.

supersample: Sets the supersample amount. Variable between 1 and 64. Default is 1.

Shift



Signal shifter with configurable input size, direction and number of shifts.

This block does not introduce extra delay.

Zeros are introduced on the shifted side.

Parameters

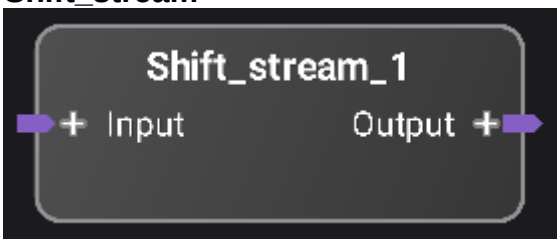
bus width: Sets the data width of the Din and Dout ports. Variable between 1 and 1024. Default is 16.

shift direction: Sets the direction to shift the Din data. Possible options are Left shift or Right shift. Default is Left shift.

shift amount: Sets the number of bits to shift. Default is 0.

supersample: Sets the supersample amount. Variable between 1 and 64. Default is 1.

Shift_stream



Signal shifter with configurable input size, direction and number of shifts.

This block does not introduce extra delay.

Zeros are introduced on the shifted side.

Parameters

bus width: Sets the data width of the Din and Dout ports. Variable between 1 and 1024. Default is 16.

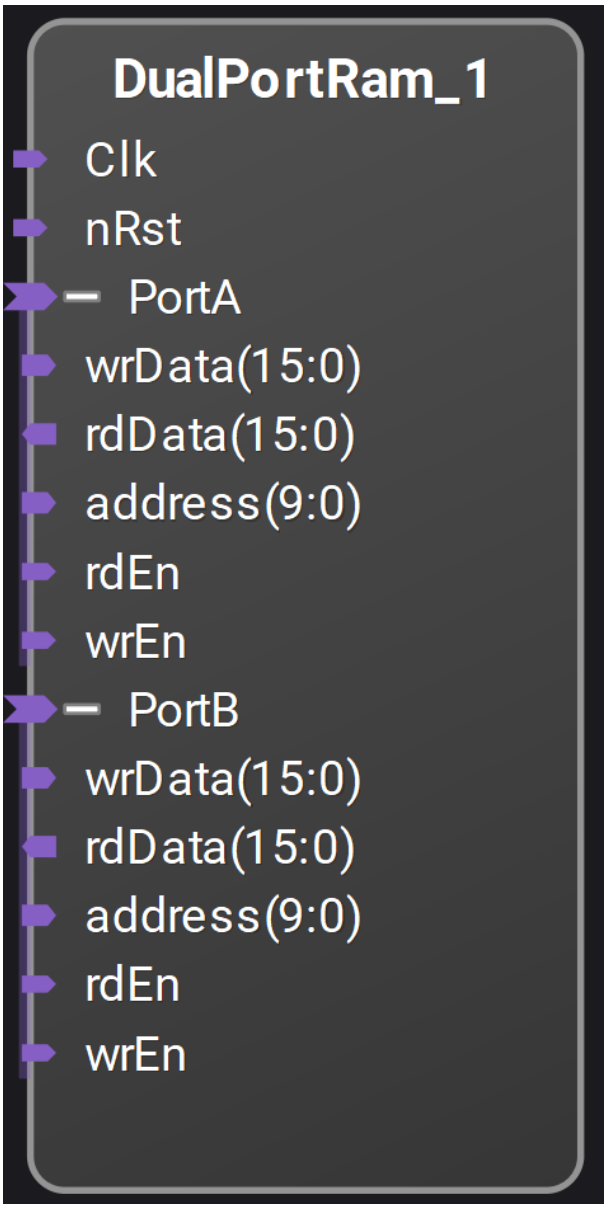
shift direction: Sets the direction to shift the Din data. Possible options are Left shift or Right shift. Default is Left shift.

shift amount: Sets the number of bits to shift. Default is 0.

supersample: Sets the supersample amount. Variable between 1 and 64. Default is 1.

Memory

DualPortRam



Dual port Block Ram up to 1024 bits x 65536 positions using PC MEM interfaces.

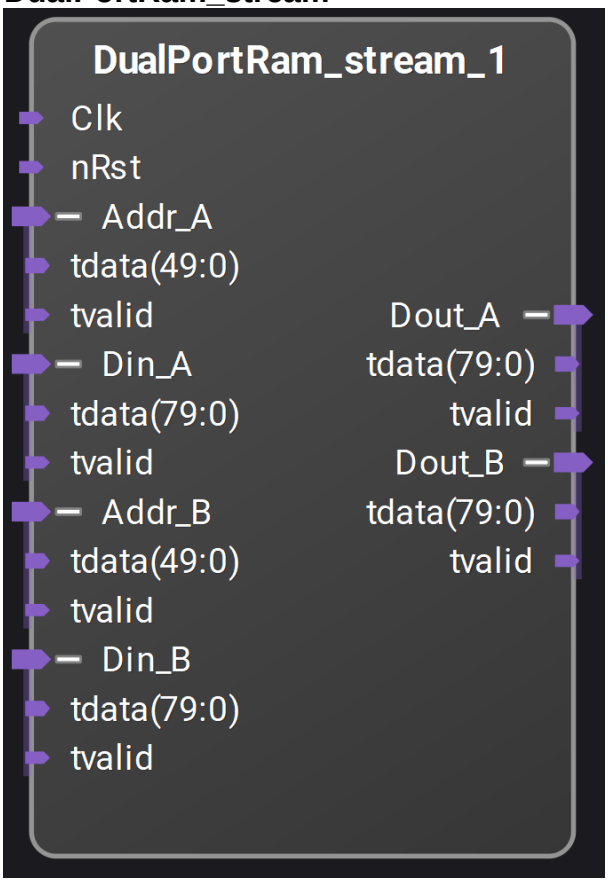
Read latency is 1 cycle.

Parameters

Data width: Sets the PortA and PortB data widths. Variable between 1 and 1024. Default is 16.

Address width: Sets the PortA and PortB address widths. Variable between 1 and 16. Default is 10.

DualPortRam_stream



Dual port Block Ram up to 1024 bits x 65536 positions using AXI Streaming interfaces.

Note that the tvalid for Addr and Din inputs must be asserted high and low at the same time for interfaces A or B.

Read latency is 1 cycle.

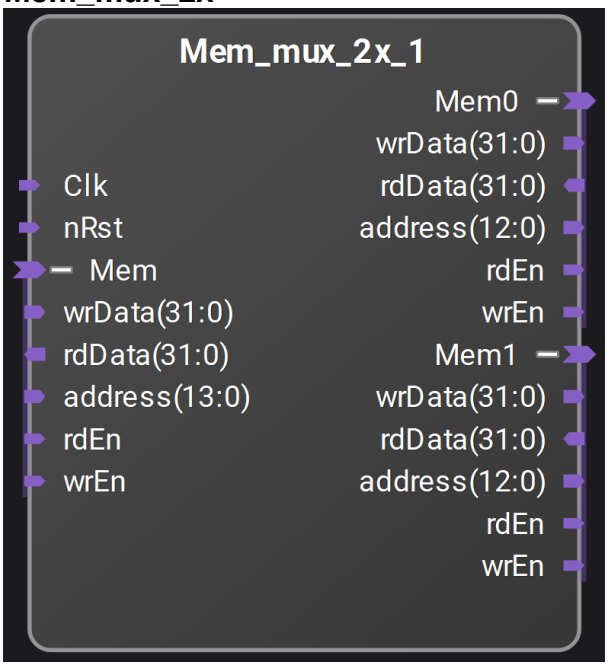
Parameters

Data width: Sets the PortA and PortB data widths. Variable between 1 and 1024. Default is 16.

Address width: Sets the PortA and PortB address widths. Variable between 1 and 16. Default is 10.

supersample: Sets the supersample amount. Variable between 1 and 64. Default is 5.

Mem_mux_2x



MEM interface 1 to 2 multiplexor.

Input address space size = 2^(Slave Address Width)

Output address space size = Input address space size / 2

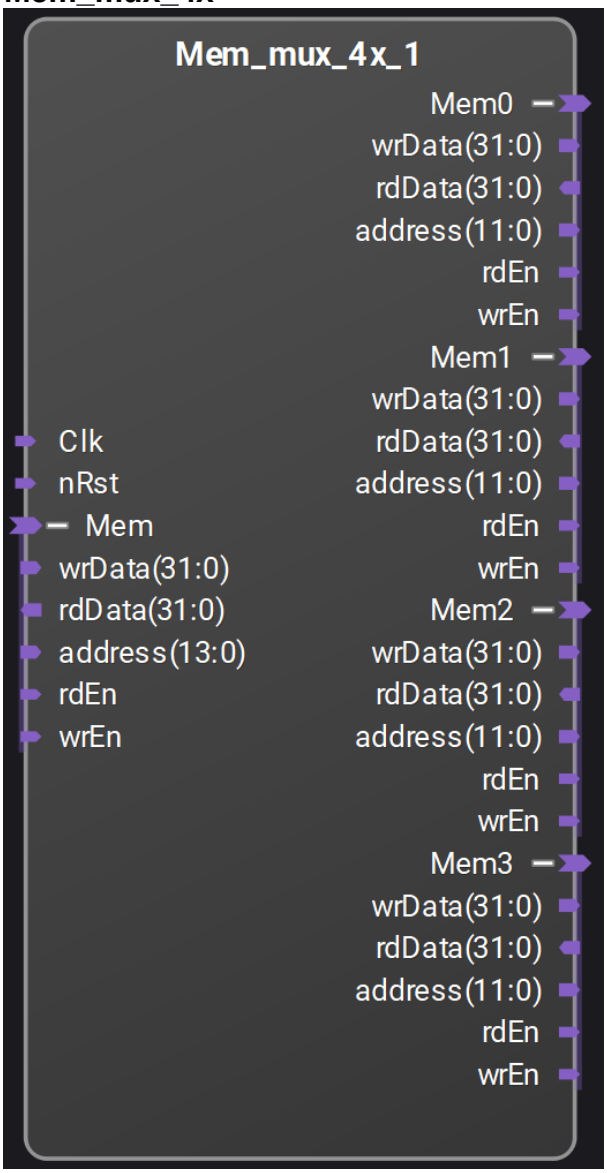
MEM0 offset = 0.

MEM1 offset = Output address space size.

Parameters

Slave Address Width: Sets the address width on the Mem interfaces. Variable between 2 and 32. Default is 14.

Mem_mux_4x



MEM interface 1 to 4 multiplexor.

Input address space size = 2^(Slave Address Width)

Output address space size = Input address space size / 4

MEM0 offset = 0.

MEM1 offset = 1*Output address space size.

MEM2 offset = 2*Output address space size.

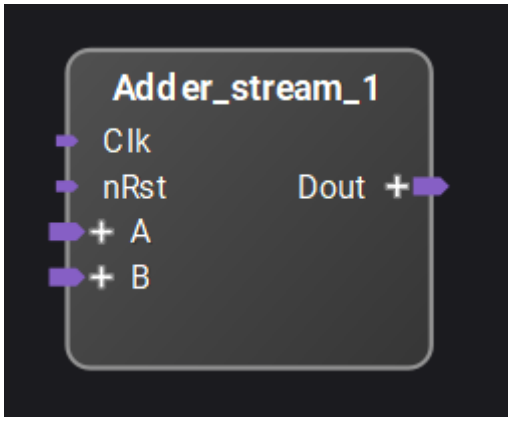
MEM3 offset = 3*Output address space size.

Parameters

Slave Address Width: Sets the address width on the Mem interfaces. Variable between 2 and 32. Default is 14.

Connecting Ports and Interfaces

Blocks can be connected together by their ports and interfaces. An interface is defined to be a set of ports.



In the example above, this block has inputs to the left (input connectors point into the block), and outputs to the right side of the block (output connectors point out of the block).

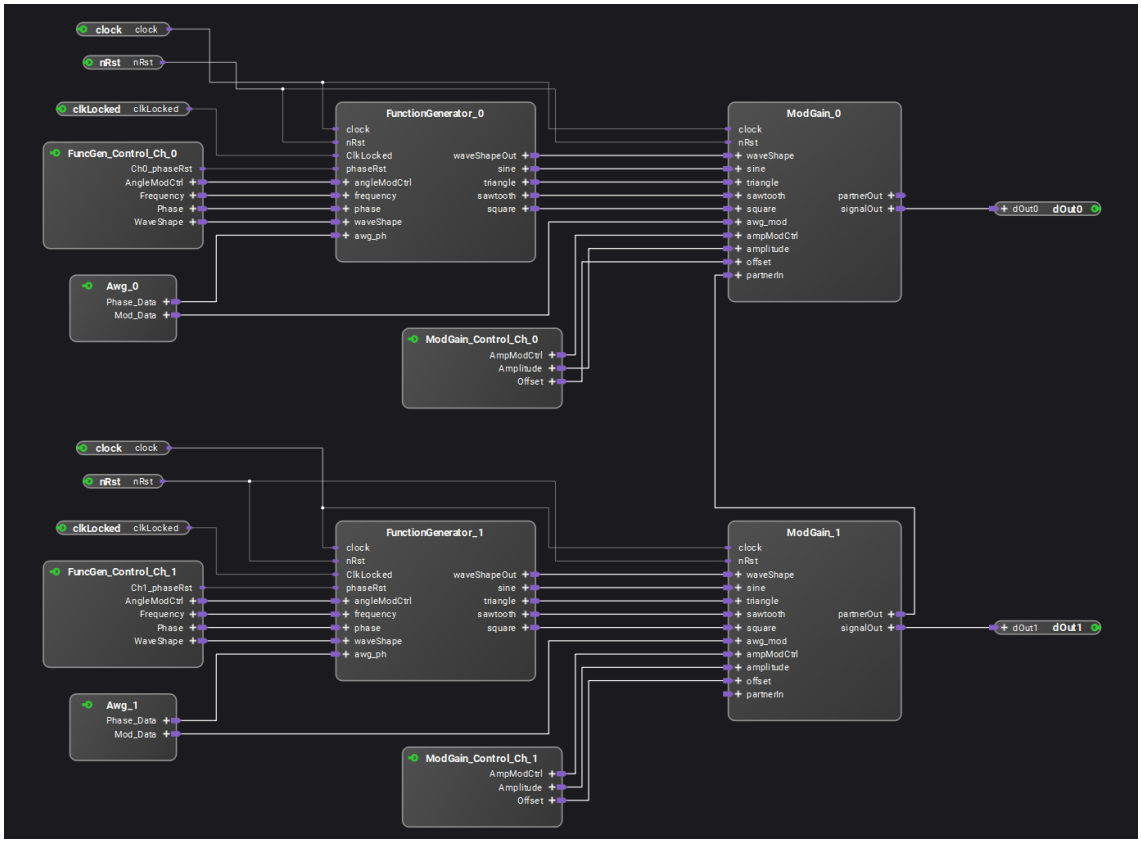
This block has two ports (small connectors), and the other connectors are interfaces (larger connectors). The ports can represent one bit of data or a vector of bits. If the port represents a vector of bits, the size can be identified next to its name.

When clicking on the "**+**" sign of an interface, such as "**A**" in the above image, the internal ports of the interface appear shown below. Notice also that the "**+**" sign has changed to a "**-**" sign. Clicking on the "**-**" sign hides the ports again.

When the "**A**" interface is connected to the output of a compatible interface, all individual signals between the two interfaces are connected. If the design requires connecting an interface to an incompatible interface or individual ports on another block, the ports within the interface may be connected instead.

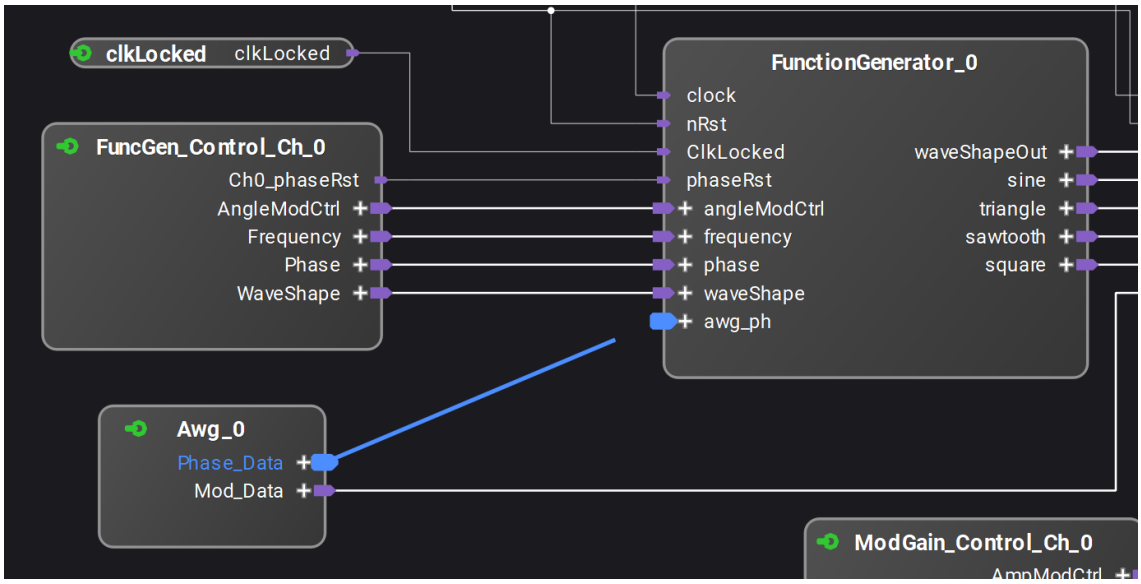
Connecting an Output Port to an Input Port

In the image below, connections are made by clicking on one port and then dragging the line from it to another suitable port. This can be done by dragging a line from an output port to an input port or by dragging a line from an input port to an output port. It may also be done by dragging a line from an input port to an existing compatible connection.

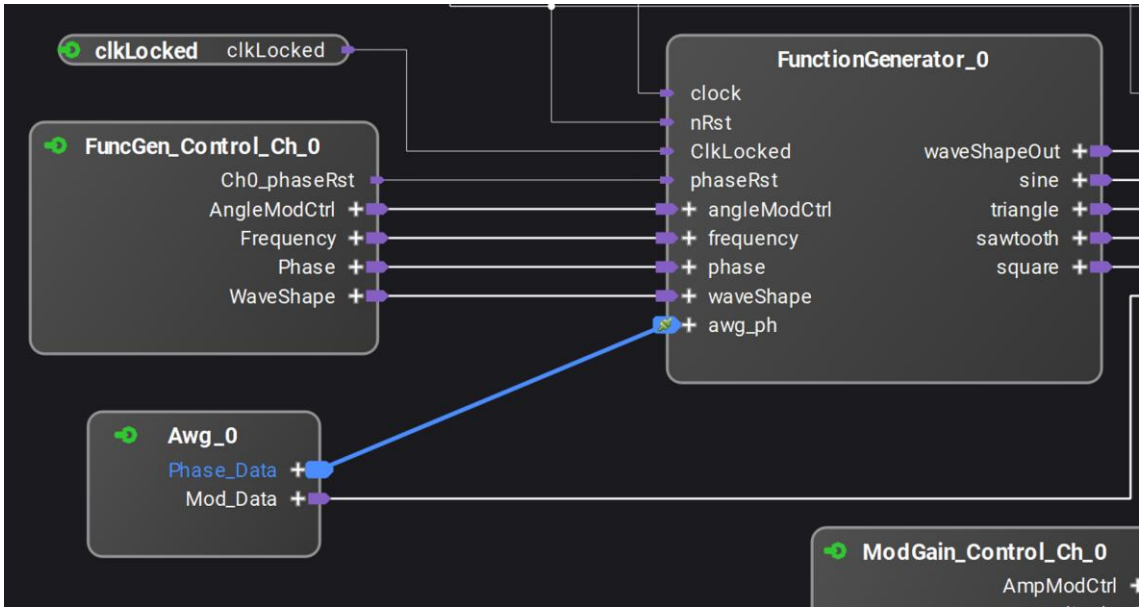


Connections can be created according to connection rules. For more information, refer Connection Rules.

If a connection can be made from a connector, a new line appears from this connector to mouse and the mouse cursor changes to the axis icon as shown below. Furthermore, the possible target connectors are highlighted in blue for showing the different connection possibilities. See the input ports on the lower block "**Awg_0**" shown below.



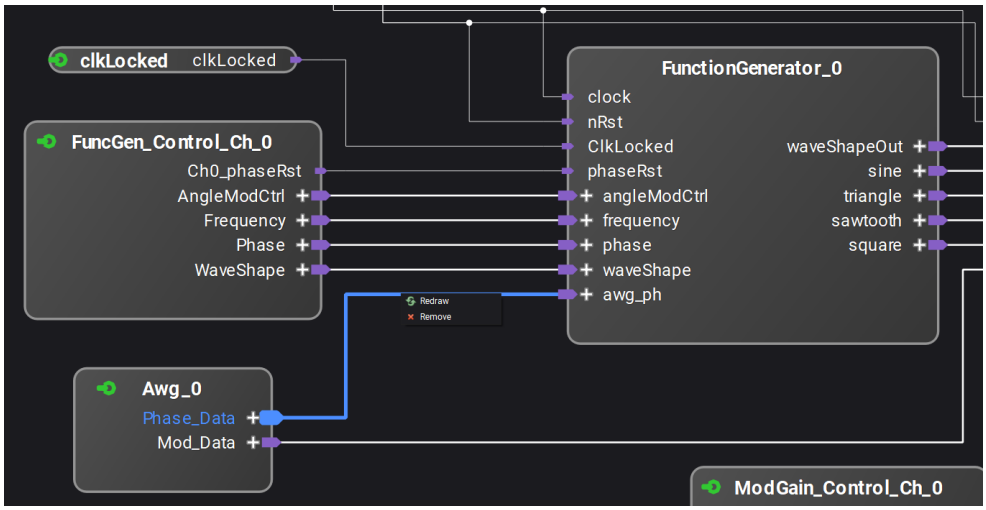
For finishing the connection, the end of the connection line is dragged by the mouse to a compatible target connector. In this case, the mouse icon changes to the green connection icon.



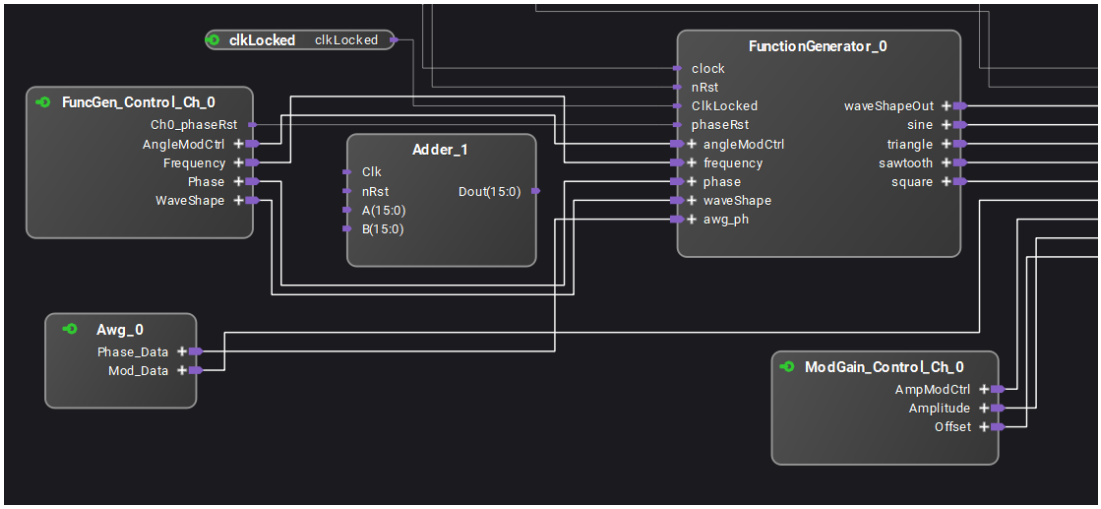
When the mouse button is released, the new connection is created.

Remove and Redraw operations

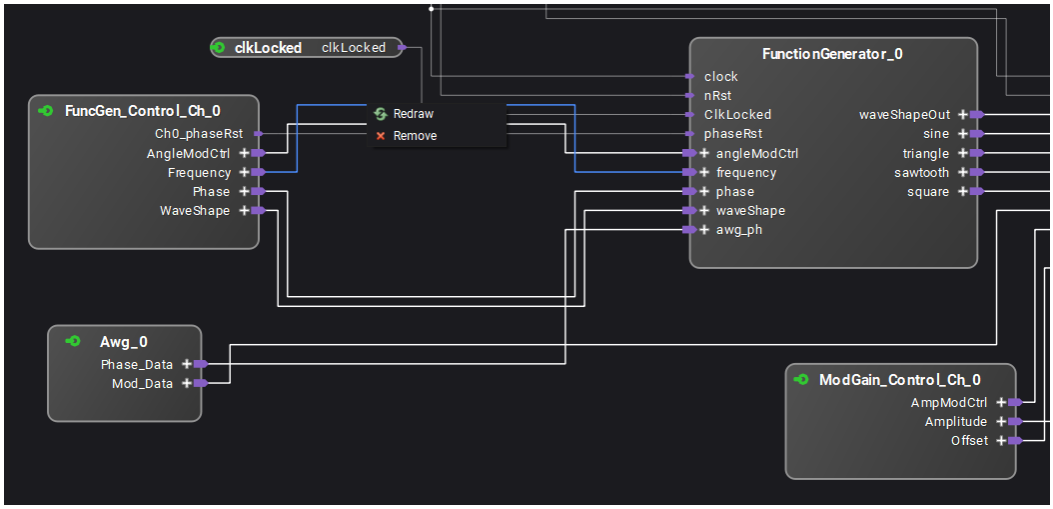
Right-click the line connecting the two ports to see two options: **Remove** and **Redraw**. Remove will delete the connecting line.



For example, add a block between the two ports. Notice the line connecting the ports is no longer straight.

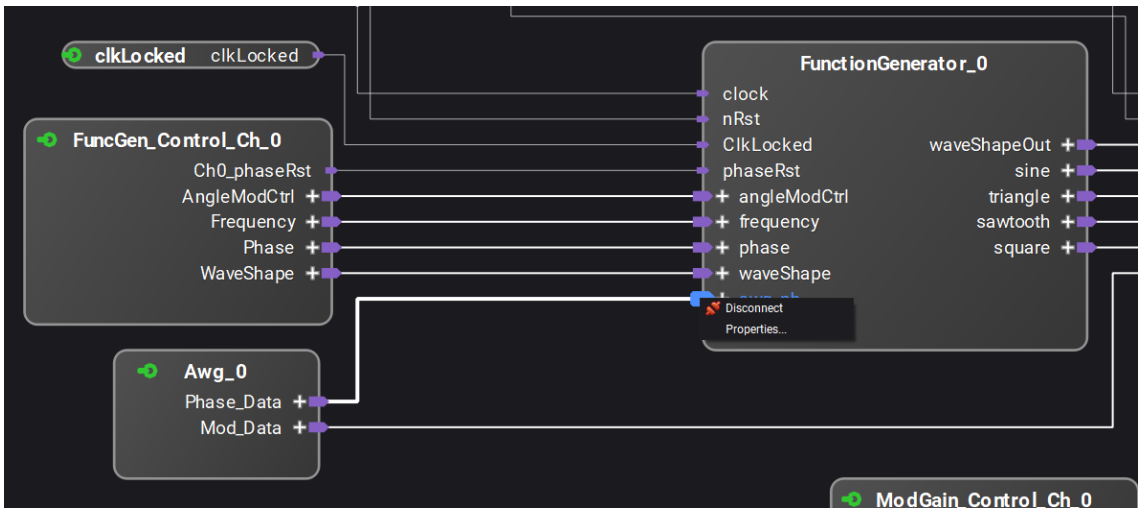


Delete the block that was just added and notice that the connecting line stays unchanged. Right-click the line and select **Redraw**. The line will be straight again.



Disconnecting a Connection

Once a connection is created, the connection can be disconnected by right-clicking on the connector, which displays the **Disconnect** option.



Connecting Input Ports to a Literal Constant

If you want to connect a input port to a constant numeric value, you should connect the port to a literal. Literals set 64-bit value constants at input ports. To insert a literal, right-click the port and select the 'Connect to literal' command. You can set the value to an integer, hexadecimal, or binary value:

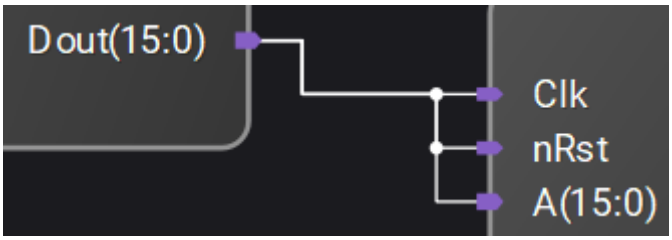
- **Integer**: A integer number, negative numbers set a two's complement format. The range for valid inputs is from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807, or from −(2^63) to 2^63 − 1
- **Hexadecimal**: A hexadecimal number using the characters 0 - F can be entered, followed by an **h**; for example, **Ah**. The range for valid inputs is from 0h to FFFFFFFFFFFFFFFFh.
- **Binary**: Binary numbers can be added followed by a **b**, for example, **1010b**.

Connection Rules

Ports

There are input ports and output ports. The input ports can have only one connection to an output port. In this example, Din(15:0) has one connection.

The output ports can be connected to multiple input ports. In this example, Dout(15:0) output is connected to three inputs.



Port Size Mismatches

If a wider output port is connected to a narrower input port, then the LSBs of the output port are used to make the connection.

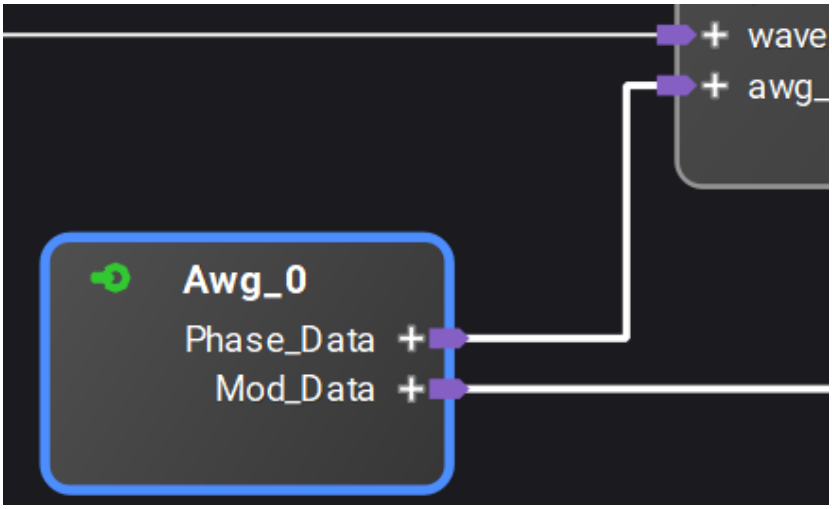
If a narrow output port is connected to a wider input port, the output port connects to the LSBs of the input port. The remaining bits of the input port are set to zero.

In general, if the smaller of the two ports has N bits, then bits N-1...0 of the output port are connected to bits N-1...0 of the input port. Any remaining output port bits are ignored, and any remaining input port bits are set to zero.

In the second example shown above, both clk and rst will be connected to Dout(0).

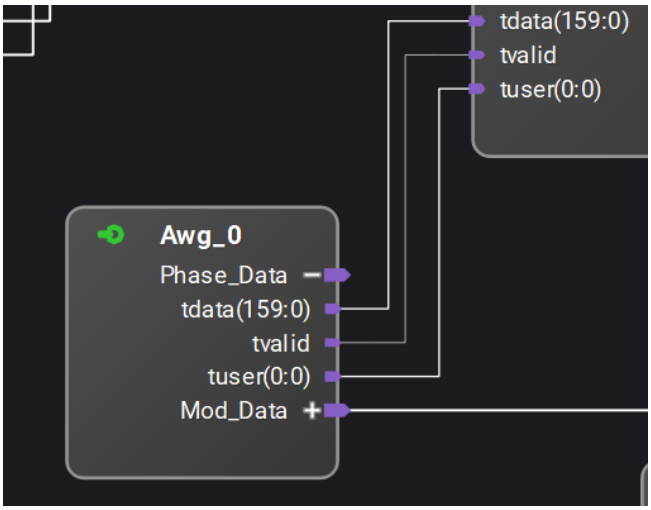
Interfaces

Interfaces with the same type can be connected. Therefore, interfaces of similar protocols can be put together with a single connection.



Clicking on the "**+**" sign for either interface will expand the interface to show the underlying ports. When an interface is expanded, clicking the "**-**" sign will collapse the port back to showing just the interface name.

Interfaces with the same number and naming of the ports can be connected together. By connecting one interface to another interface, as shown above, all the corresponding ports shown are connected. This removes the chore of having to connect each interface port as shown below.



Naming Conventions

Within PathWave FPGA, things like Instance names and Register names must be unique and valid HDL identifiers. Specifically they must follow these rules:

1. A name must start with an alphabetic character (A-Z or a-z).
2. A name can only consist of of alphanumeric characters and underscores (A-Z, a-z, 0-9, _).
3. A name must end with an alphanumeric character (A-Z, a-z, 0-9).
4. A name can not be a reserved word (listed below).
5. Names are not case sensitive. Thus *myreg*, MY*REG, MyReg* are all considered the same.

Reserved Words

The following are reserved words and can not be used as names:

```
abs, access, after, alias, all, always, always_comb, always_ff,
always_latch, and, architecture, array, assert, assign, assume,
```

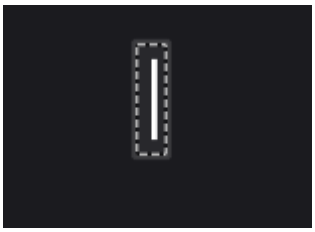
```
attribute, automatic, before, begin, bind, bins, binsof, bit, block,
body, break, buf, buffer, bufif0, bufif1, bus, byte, case, casex,
casez, cell, chandle, class, clocking, cmos, component, config,
configuration, const, constant, constraint, context, continue, cover,
covergroup, coverpoint, cross, deassign, default, defparam, design,
disable, disconnect, dist, do, downto, edge, else, elsif, end,
endcase, endclass, endclocking, endconfig, endfunction, endgenerate,
endgroup, endinterface, endmodule, endpackage, endprimitive,
endprogram, endproperty, endsequence, endspecify, endtable, endtask,
entity, enum, event, exit, expect, export, extends, extern, file,
final, first_match, for, force, forever, fork, forkjoin, function,
generate, generic, genvar, group, guarded, highz0, highz1, if, iff,
ifnone, ignore_bins, illegal_bins, import, impure, in, incdir,
include, inertial, initial, inout, inout, input, inside, instance,
int, integer, interface, intersect, is, join, join_any, join_none,
label, large, liblist, library, linkage, literal, local, localparam,
logic, longint, loop, macromodule, map, matches, medium, mod, modport,
module, nand, negedge, new, next, nmos, nor, nor, noshowcancelled,
not, notif0, notif1, null, of, on, open, or, others, out, output,
package, packed, parameter, pmos, port, posedge, postponed, primitive,
priority, procedure, process, program, property, protected, pull0,
pull1, pulldown, pullup, pulsestyle_ondetect, pulsestyle_onevent,
pure, rand, randc, randcase, randsequence, range, rcmos, real,
realtime, record, ref, reg, register, reject, release, rem, repeat,
report, return, rnmos, rol, ror, rpmos, rtran, rtranif0, rtranif1,
scalared, select, sequence, severity, shared, shortint, shortreal,
showcancelled, sig, signal, signed, sla, sll, small, solve, specify,
specparam, sra, srl, static, string, strong0, strong1, struct,
subtype, super, supply0, supply1, table, tagged, task, then, this,
throughout, time, timeprecision, timeunit, to, tran, tranif0, tranif1,
transport, tri, tri0, tri1, triand, trior, trireg, type, typedef,
unaffected, union, unique, units, unsigned, until, use, uwire, var,
variable, vectored, virtual, void, wait, wait_order, wand, weak0,
weak1, when, while, wildcard, wire, with, within, wor, xnor, xor
```

Adding and Editing Comments

To add a comment:

1. Position the cursor within the project where the comment is to be inserted.

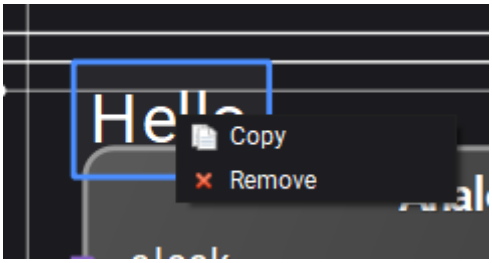
2. Right-click on a blank part of the canvas and select **Insert Comment...** .



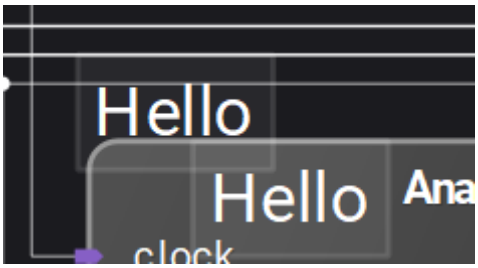
3. Add text to the comment text box.

4. The comment can be moved by dragging it with the mouse. Notice the comment is in the foreground and appears above the project elements.

5. On right-clicking the comment, the option to copy or remove is provided.



6. Choose **Copy**, to create a duplicate comment.



7. Choose **Remove**, to delete the comment.

Naming Collisions

PathWave FPGA is using the concept of VLNV for identifying IP and reporting naming collisions. VLNV stands for Vendor-Library-Name-Version and is a concept introduced by IP-XACT.

- **Two IPs have the same name, but different VLNV.** In this case, user will have to resolve it using one of the workarounds.
- **Two IPs have have the same VLNV, apart from the version field.** In this case, PathWave FPGA will give the user the option to upgrade/downgrade. Note that this option is not available if the IPs are coming from an IP repository. In the latter case, user will have to resolve it using one of the workarounds.
- **Two IPs have the same VLNV, but different contents.** In this case, PathWave FPGA will give the user the option to update to the desired definition. Note that this option is not available if the IPs are coming from an IP repository. In the latter case, user will have to resolve it using one of the workarounds.
- **Two IPs have the same VLNV and contents, but are stored in different location.** In this case, PathWave FPGA will use the last loaded location as the correct location of the IP.
- **Two IPs have the same name, but they do not have a VLNV.** In this case, user will have to resolve it using one of the workarounds.
- **Two IPs have the same name, but are coming from different import method.** In this case, user will have to resolve it using one of the workarounds.
- **An IP is using a name of a reserved word.** In this case, a possible workaround is to create a wrapper for that IP which will have a non-colliding name

Workarounds

When a name collision is detected, the user will have to take action and resolve it.

- **Rename the IP to a non-conflicting name.** This is simplest and fastest solution. However, if the user is not the owner of the IP, it might not be feasible. In this case, the user has to follow the second workaround
- **Load only the IPs that are necessary for the project.** This is by definition possible only if the conflicting IPs are not needed at the same time in the design. If they are, the previous workaround in the only option. Note that in the case of unwanted IPs that are loaded through an IP Repository location, user has to either remove the IP Repository location, which will also remove any other IP loaded from the same place, or, if this is not possible, move the conflicting IP definition file (IP-XACT file) outside of the IP repository location or any sub-directory.
- **Create a wrapper entity/module for the failing IP.** This option will only work if the reason of the name collision is a reserved word or the name of the IP matches the name of a sandbox interface. The wrapper entity has to use a non-conflicting name.

Generating the Bit File

- Synthesizing and Implementing your Design inside of PathWave FPGA
    - Monitoring the Build
    - Exploring the Build Output
- Building your Design using Vivado

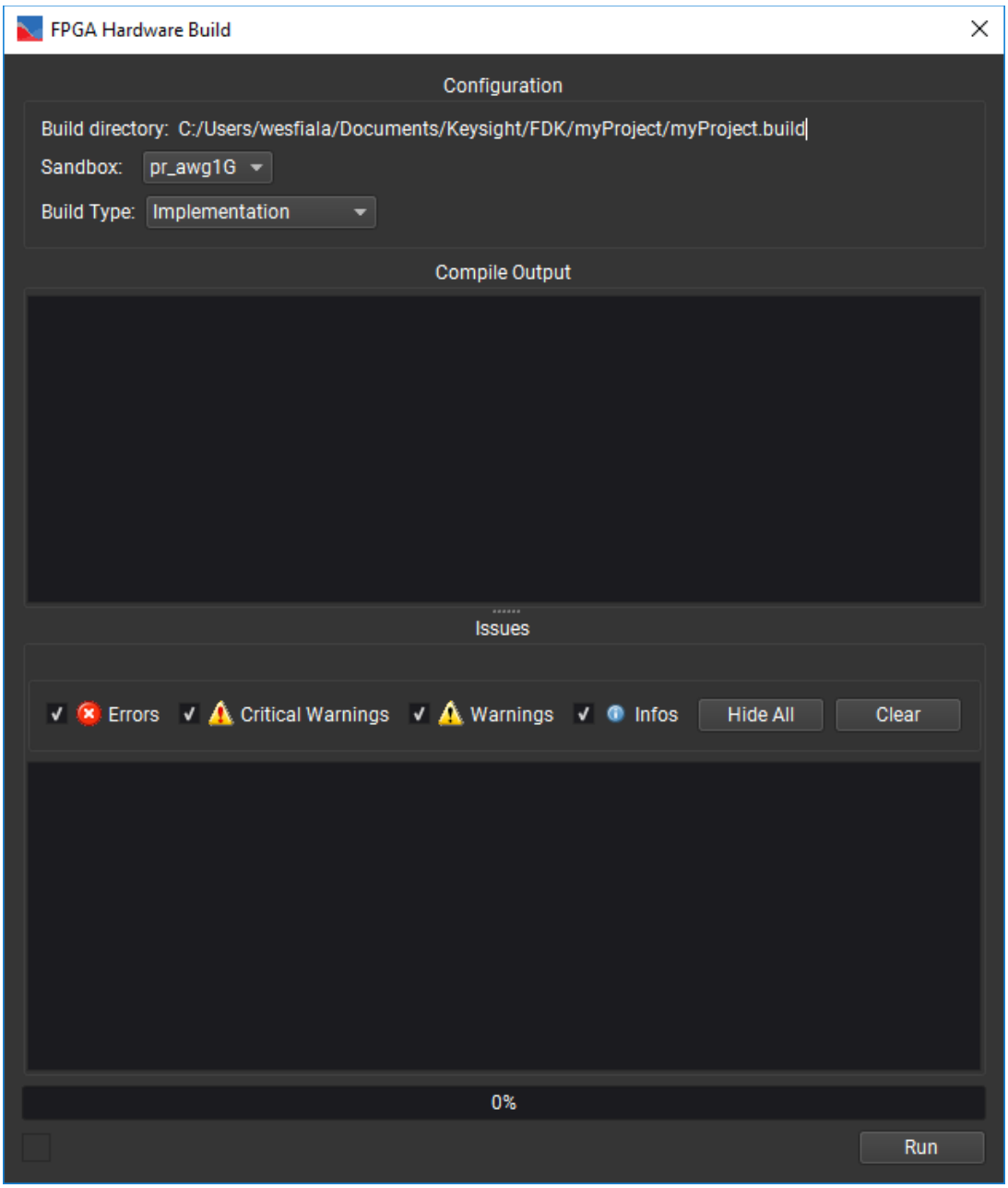
- Generating a Vivado Project
- Building your Vivado Project
  - Implementating from PathWave FPGA
  - Building Entirely in Vivado

Synthesizing and Implementing your Design inside of PathWave FPGA

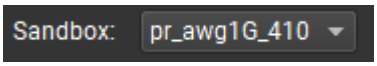
After creating your new hardware project and adding your FPGA logic, you are ready to generate the bit file that implements your design.

To build the bitfile based on your design, complete the following steps:

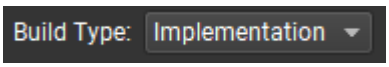
1. Select **Module**> **Generate Bit File...** or click the toolbar icon with tooltip "Generate Bit File...". The FPGA Hardware Build dialog will appear.



2. Choose the sandbox that you want to target for this build.



3. Choose the **Implementation** build type. This will build the complete project, including the bit file.



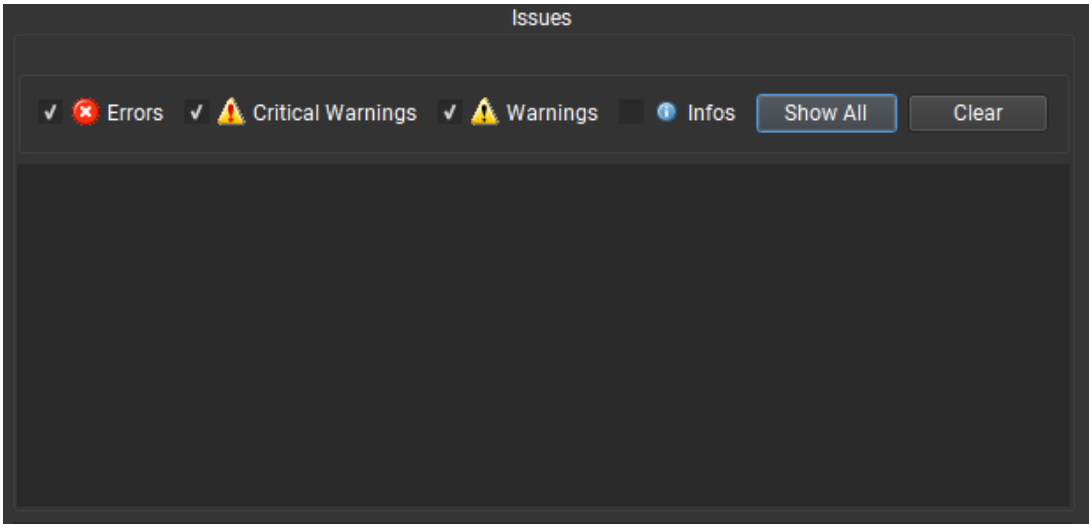
4. Click **Run** to start the build.

Monitoring the Build

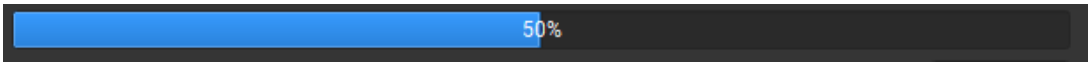
The FPGA Hardware Build dialog contains several panes to monitor the progress of the build:

- The Compile Output pane displays all build output.

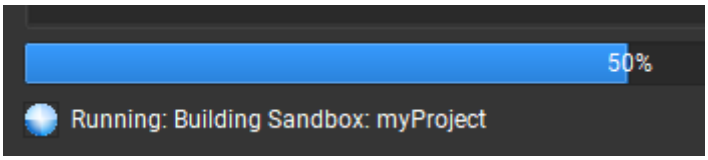
- The Issues pane shows filtered build output. You can set the filters by checking the boxes (Errors, Critical Warnings. etc.) at the top of the Issues pane. The filters can be set at any time while the build is running or after it is complete.



- The progress bar shows the approximate progress of the build.

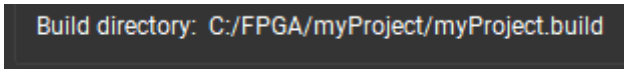


- The status bar at the bottom left shows what step of the build is being performed. When the build is finished, the build status will be displayed.



Exploring the Build Output

The Build directory field in the Configuration pane specifies the parent directory of the build artifacts, including the generated bit file. The Program Archive of the generated bit file may be recognized by its k7z file extension.



If the build was successful, the build artifacts are copied to an artifact directory for future reference. Each set of build artifacts has its own time and date stamped directory. In this example, one artifact directory could be named myProject.data\bin\myProject_2018-04-04T14_21_55.

To learn more about the build output structure, refer to the Project Directory Structure section.

Building your Design using Vivado

A native Vivado flow for users who want to use advanced features in Vivado (such as adding placement constraints).

Generating a Vivado Project

To start the advanced build flow and leave PathWave FPGA build environment, follow the steps listed below.

1. Open a new/existing PathWave FPGA project, and navigate to the FPGA Hardware Build dialog.
2. Select the sandbox you wish to implement with the sandbox drop down, and select the **Project Generation** build type.

3. Hit **Run**.
    a. If any build errors are encountered, solve the errors before continuing.
4. Find the generated Vivado Project at {Project Folder}/{Project Name}.build/{Project Name}_Generated_Project.
    a. The project generation flow will not overwrite an existing generated project and will instead insert a counter of the form '_{iter}' to the name of the generated project. i.e. myProject_Generated_Project_1.
5. Open the generated project.

A Vivado project is now created and ready for development. The following sections will discuss how to finish implementation after development.

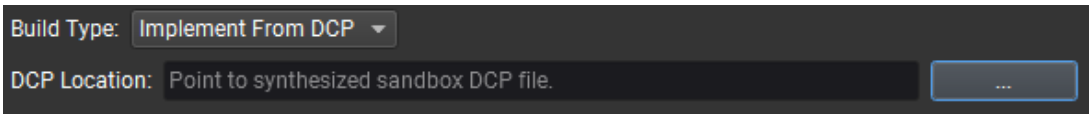
Building your Vivado Project

Once development is finished on the sandbox, the user takes an out of context synthesized DCP and reintroduces it into the PathWave FPGA build. A separate flow that does not restart PathWave FPGA is also available.

Implementating from PathWave FPGA

In this flow, the user takes their developed sandbox and creates an out of context synthesized DCP and passes it into PathWave FPGA to create the project outputs. Below are the steps for this flow.

1. Open the Vivado project that was created for the chosen sandbox, that has finished development.
2. Run the following tcl command in Vivado: **synth_design -mode out_of_context**
3. Run the following tcl command in Vivado: **write_checkpoint ${Location you wish to put DCP}**
4. Open the PathWave FPGA project that was used to create the Vivado Project.
5. Navigate to the **FPGA Hardware Build** dialog**.**
6. In the **Build Type** combo box select **Implement from DCP**.



7. Either manually enter the location of your sandbox DCP file, or click the browse button and select the DCP file.
8. Click **Run**.

PathWave FPGA will take the out of context synthesized DCP and run its implementation script. All build messages will be displayed like normal, and the output of the build will produced as is above in the normal flow.

Building Entirely in Vivado

In this flow, the user takes their developed sandbox and manually runs the build scripts provided by the BSP. Below are the steps for this flow.

1. Open Vivado, but do not open a project.
2. Locate the build script for the BSP that the sandbox is designed around. The recommended storage location is in {BSP Location}/fsp/script.
    a. Although not necessary for implementation, it is recommended to read through this build script to understand what is going on.
3. Run the following tcl command in Vivado: **source ${BSP Build Script}**
    a. This adds some tcl commands into Vivado.
4. Run the following tcl command in Vivado: **::Keysight::GT::ImplSandboxes ${args}**
    a. The arguments for this command are listed below (case sensitive):
        i. -pn : project name
        ii. -n : number of sandboxes
        iii. -fpga : FPGA part number
        iv. -k${sbx}n : name of the kernel to be loaded to sandbox ${sbx}
        v. -k${sbx}p : path of the dcp checkpoint of the kernel to be loaded to sandbox ${sbx}

vi. -k${sbx}c : clock frequency of the kernel to be loaded to sandbox ${sbx}

vii. -sdcp : static dcp file

viii. -bit : bit filepath

ix. -P : project directory

x. -fsp : BSP configuration filepath (.fspinfo)

b. Reading through the .fspinfo file for the BSP you are building for may help you find the values you need to use for these arguments.

After running the above tcl proc, a design has been created and can be edited as the user requires. If the build completed successfully, the project outputs will be produced and ready to be deployed.

Verifying the Bit File

After you generate your FPGA bit file, you are ready to deploy and verify it on the FPGA. The Board Support Package for your FPGA supplies the *run-time support package* (RSP) C API that provides programmatic control of the FPGA. Using the RSP you can create a C application to verify your bit file. Note, you will need Visual Studio C++ and CMake, please see the System Requirements for more details.

The RSP documentation and example program are provided in a separate Help area available from the **Help > Programmer's Guide** menu.

After you have verified the bit file, you are ready to deploy it in a measurement application. Please consult your instrument driver manual to learn how to integrate the bit file into your custom measurement application.

Glossary

Term	Definition
Bit file	File built from the user design containing the bits to download to the FPGA sandbox.
Block	An HDL IP block that is placed on the PathWave FPGA design schematic.
Board support package (BSP)	A package containing all of the necessary content to target a Keysight Open FPGA. These are installed separately from PathWave FPGA. A BSP is made up of two parts, the *FPGA support package* (FSP) and the *run-time support package* (RSP) .
FPGA support package (FSP)	The portion of the BSP that allows you to build a bit file for the target FPGA.
Interface	A set of ports for a block that can be connected to another compatible interface. Alternatively, an interface can be expanded and the individual ports can be connected to another compatible port.
Module	Either a top level module or submodule that is currently the top level module for simulation purposes
Port	An input or output signal to a block.
Program archive	An archive file (.k7z) containing one or more bit files and associated metadata.

Term	Definition
Run-time support package (RSP)	The portion of the BSP that allows you to control your target FPGA. It provides a C API that you can use to download and verify your FPGA bit image.
Sandbox	The user-configurable region in the FPGA.
Submodule	Hierarchical schematic design that can be instantiated in either a top level module or another submodule
Top level module	Top of the user design, defines the IO of the sandbox.

IP Developers Guide

This IP developers guide describes how an IP block must be packaged to be included in PathWave FPGA. This guide also describes IP restrictions and how the IP restrictions are formatted in the IP-XACT XML file. IP restrictions determine which FPGA vendors (eg. Xilinx) along with which FPGA families (eg. Virtex 7) and BSPs (eg. M3202A) are supported.

IP Repositories

IP repositories are directories that contain all the artifacts required to describe an IP. For an IP to be discovered by PathWave FPGA, an IP-XACT file (of the IEEE 1685-2014 standard) is required. The role of the IP-XACT file is crucial to identify an IP, represent its interfaces, and describe all its resources (source, constraint, documentation, simulation files). In other words, an IP-XACT file will fully define an IP and describe its directory structure. To load an IP repository, use the Settings Dialog.

IP directory structure

The directory structure for an IP is left up to the IP developer to define. However, a proposed directory structure, which is similar to the one used from Xilinx Vivado, is the following:

- MyIPlibrary ← IP library top level directory
    - doc ← IP documentation
    - src ← IP HDL source directory. Source code may be encrypted. This directory contains both the behavioral and synthesizable code.
    - tb ← A directory for the IP simulation testbench.
    - MyIPlibrary.xml ← An IP-XACT file that describes the IP

Definition of the IP-XACT file

As described above, PathWave FPGA will identify IPs that exist in an IP repository directory by discovering the IP-XACT files that describe those IPs. For an IP-XACT file to correctly identify an IP, the guidelines described below should be followed:

- **the IP-XACT file should follow the IEEE 1685-2014 standard**
- **the root element should be an ipxact:component**
- **the vendor name (element ipxact:vendor, first child of ipxact:component) should be equal to the internet domain of the vendor of the IP (for example, for Keysight Technologies this will be keysight.com)**
- **the name of the library (element ipxact:library, first child of ipxact:component) will be the name of the library the IP belongs to. This name is also used inside PathWave FPGA for categorizing the IPs**
- **the name (element ipxact:name, first child of ipxact:component) should be the same as the name of the IP (*module name* in Verilog, SystemVerilog and SystemC, or *entity name* in VHDL)**
- **if the IP uses Keysight Standard Interfaces, these should be described using ipxact:busInterface elements**
- **the mappings between logical ports of the 'busInterface's to the physical ports of the IP should be '1 to 1'. This means that one physical port maps completely (same width, direction) and only to one logical port**
- **the files that are necessary for an IP to be included in a build process (synthesis, implementation, bit generation) should be defined inside an ipxact:fileset component, named "synthesis".**

A detailed description of all the elements that are required by PathWave FPGA in order to identify correctly an IP is given in the following table. For more information on the various elements that are supported by IP-XACT, please consult the manual IEEE 1685-2014 standard.

Element	Parent Element	Content
ipxact:component	<root>	This is the root element of the XML file
ipxact:vendor	ipxact:component	Vendor's name. Should be equal to the internet domain of the vendor of the IP (e.g. keysight.com)
ipxact:library	ipxact:component	The name of the library the IP belongs to
ipxact:name	ipxact:component	The name of the IP. Should be the same as the name of the IP in the source file (i.e. *module name* in Verilog, SystemVerilog and SystemC, or *entity name* in VHDL)
ipxact:version	ipxact:component	The version number of the IP.
ipxact:busInterfaces	ipxact:component	Contains a list of ipxact:busInterface elements
ipxact:busInterface	ipxact:busInterfaces	Contains information about a used Keysight Standard Interface
ipxact:name	ipxact:busInterface	The name of the Interface that is used in this IP
ipxact:busType	ipxact:busInterface	The type of the Interface that is used in this IP. This essentially is the VLNV of the

Element	Parent Element	Content
		Keysight Standard Interface to be used. This should match one of the bus definitions (IP-XACT files with <ipxact:busDefinition> as the root element) defined by PathWave FPGA. See Keysight Standard Interfaces for more information
ipxact:abstractionTypes	ipxact:busInterface	Contains a list of ipxact:abstractionType elements. PathWave FPGA will only support one, the first
ipxact:abstractionType	ipxact:abstractionTypes	Contains information about a used Keysight Standard Interface and the mapping to the physical ports
ipxact:abstractionRef	ipxact:abstractionType	The type of the Interface definition that is used in this IP. This essentially is the VLNV of the definition of the Keysight Standard Interface to be used. This should match one of the abstraction definitions (IP-XACT files with <ipxact:abstractionDefinition> as the root element) defined by PathWave FPGA. See Keysight Standard Interfaces for more information
ipxact:portMaps	ipxact:abstractionType	Contains a list of ipxact:portMap elements
ipxact:portMap	ipxact:portMaps	Contains information about a specific port mapping
ipxact:logicalPort	ipxact:portMap	Contains information about the logical port (port defined in the abstractionDefinition of the enclosing abstractiontype) that participates in the port mapping
ipxact:name	ipxact:logicalPort	The name of the logical port (As this is defined in the abstractionDefinition for the selected Interface Type)
ipxact:physicalPort	ipxact:portMap	Contains information about the physical port (port of the IP) that participates in the port mapping
ipxact:name	ipxact:physicalPort	The name of the physical port (As this is defined in the ipxact:ports section in the same file)
ipxact:model	ipxact:component	Contains information about the modeling of the IP
ipxact:ports	ipxact:model	Contains a list of ipxact:port elements, which represent the physical ports of the IP
ipxact:port	ipxact:ports	Contains information about a specific physical port
ipxact:name	ipxact:port	The name of the physical port. This should match the name defined in the source HDL file

Element	Parent Element	Content
ipxact:wire	ipxact:port	Contains information about the physical representation of a physical port
ipxact:direction	ipxact:wire	Specifies the direction of this port: in for input ports, out for output ports
ipxact:vectors	ipxact:wire	Contains a list of ipxact:vector elements. PathWave FPGA will only support one, the first
ipxact:vector	ipxact:vectors	Specifies the dimensions for a non-scalar port
ipxact:left	ipxact:vector	Specifies the left range for the bit slice used to map a port vector to the bus interface
ipxact:right	ipxact:vector	Specifies the right range for the bit slice used to map a port vector to the bus interface
ipxact:fileSets	ipxact:component	Contains a list of ipxact:fileSet elements
ipxact:fileSet	ipxact:fileSets	Contains information about a specific set of files. Can contain one or multiple ipxact:file elements
ipxact:name	ipxact:fileSet	The name for this set of files.
ipxact:file	ipxact:fileSet	Contains information about a specific file
ipxact:name	ipxact:file	The path to the file. This should be relative to the path of the current IP-XACT document
ipxact:fileType	ipxact:file	Describes the type of file. PathWave FPGA understands one of the following names: • vhdlSource: It is a VHDL source file • verilogSource: It is a Verilog source file • systemVerilogSource: It is a SystemVerilog source file • user: It is a user defined source, described by the attribute "user"
user	attribute of ipxact:fileType	Can be: • xci: Xilinx Core Instance • dcp : It is a Vivado design checkpoint file
ipxact:description	ipxact:component	A short description of the IP

Keysight Standard Interfaces

The bus interfaces that are currently supported by PathWave FPGA to be used inside an IP component definition are described as Keysight Standard Interfaces. Each of these interfaces has IP-XACT definitions, which are defined by, and installed with, PathWave FPGA.

More specifically, for each interface, two IP-XACT files are defined:

- the **Bus Definition**: IP-XACT file with ipxact:busDefinition as root element
- the **Abstraction definition**: IP-XACT file with abstractionDefinition as root element

The **Bus Definition** is used to define the high-level details of an interface, such as if is addressable or not, if it supports direct connection between a master and a slave, etc.

Code Block 2 Example Bus Definition for AXI4-Stream interface

```
<ipxact:busDefinition xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-
2014" xsi:schemaLocation="http://www.accellera.org/XMLSchema/IPXACT/1685-
2014/http://www.accellera.org/XMLSchema/IPXACT/1685-2014/index.xsd">
 <ipxact:vendor>keysight.com</ipxact:vendor>
 <ipxact:library>interfaces</ipxact:library>
 <ipxact:name>axis</ipxact:name>
 <ipxact:version>1.0</ipxact:version>
 <ipxact:directConnection>true</ipxact:directConnection>
 <ipxact:isAddressable>false</ipxact:isAddressable>
</ipxact:busDefinition>
```

The **Abstraction Definition** is used to define the low-level details of an interface, such as the port and the parameter list.

Code Block 3 Example Abstraction Definition for AXI4-Stream interface

```
<ipxact:abstractionDefinition
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014"
xsi:schemaLocation="http://www.accellera.org/XMLSchema/IPXACT/1685-
2014/http://www.accellera.org/XMLSchema/IPXACT/1685-2014/index.xsd">
 <ipxact:vendor>keysight.com</ipxact:vendor>
 <ipxact:library>interfaces</ipxact:library>
 <ipxact:name>axis.absDef</ipxact:name>
 <ipxact:version>1.0</ipxact:version>
 <ipxact:busType vendor="keysight.com" library="interfaces" name="axis"
version="1.0"/>
 <ipxact:ports>
    <ipxact:port>
       <ipxact:logicalName>tdata</ipxact:logicalName>
       <ipxact:wire>
          <ipxact:qualifier>
             <ipxact:isData>true</ipxact:isData>
          </ipxact:qualifier>
          <ipxact:onMaster>
             <ipxact:presence>optional</ipxact:presence>
             <ipxact:width>64</ipxact:width>
             <ipxact:direction>out</ipxact:direction>
          </ipxact:onMaster>
          <ipxact:onSlave>
             <ipxact:presence>optional</ipxact:presence>
             <ipxact:width>64</ipxact:width>
             <ipxact:direction>in</ipxact:direction>
          </ipxact:onSlave>
          <ipxact:defaultValue>0</ipxact:defaultValue>
       </ipxact:wire>
    </ipxact:port>
    <ipxact:port>
       <ipxact:logicalName>tvalid</ipxact:logicalName>
       <ipxact:wire>
          <ipxact:onMaster>
             <ipxact:presence>required</ipxact:presence>
             <ipxact:width>1</ipxact:width>
```

```
            <ipxact:direction>out</ipxact:direction>
         </ipxact:onMaster>
         <ipxact:onSlave>
            <ipxact:presence>required</ipxact:presence>
            <ipxact:width>1</ipxact:width>
            <ipxact:direction>in</ipxact:direction>
         </ipxact:onSlave>
      </ipxact:wire>
   </ipxact:port>
      :
      :
      :
 </ipxact:ports>
</ipxact:abstractionDefinition>
```

Managing Multiple Clocks and Resets

PathWave FPGA needs to know which clock synchronous interfaces use. If there is only one clock in an IP block's definition, then there is no ambiguity. However, if there is more than one clock interface, then the tools need to know which clock corresponds to which interfaces. To do this, one adds the **ASSOCIATED_BUSIF** parameter to the bus interface definition of each clock interface. The value of the **ASSOCIATED_BUSIF** parameter is a colon separated list of the names of the interfaces that use that clock. This should include all the synchronous interfaces (things such as AXI and PC_MEM interfaces) that use that clock. If every synchronous interface uses the same clock, then the **ASSOCIATED_BUSIF** can be set to the value ***** as a wildcard to denote all interfaces.

Additionally, reset signals are usually synchronous with a clock in order to generate clean reset events. If there are more than one clock and more than one reset signal, then PathWave FPGA also needs to know which reset signal is associated with a particular clock. To do this, one adds the **ASSOCIATED_RESET** parameter to the bus interface definition of the pertinent clock interface. The value of the **ASSOCIATED_RESET** parameter is the name of a single reset interface that should be used with that clock. Note that while **ASSOCIATED_BUSIF** can accept multiple colon separated names or the * wildcard, **ASSOCIATED_RESET** can only be a single name.

Parameterizing IP Designs

For added generality, IP-XACT standard allows the usage of *parameters* to control various aspects of the IP block's definition, so that the same block may be used with different configurations. These parameters can be simple constants such as 16, or they can be mathematical expressions involving multiple constants and/or other parameters. The format of expressions in IP-XACT are detailed in *Annex C* of the IP-XACT 1685-2014 standards document. The format is based on *System Verilog*'s expression syntax.

IP-XACT provides different ways to define parameters, however, in the context of Pathwave FPGA, two methods are currently supported:

- **Component Parameters**

- **Module Parameters**

The following table summarizes the elements/attributes that PathWave takes into account when parsing an IP-XACT file, with respect to the parameters:

Element/Attribute	Parent Element	Content
ipxact:parameter	ipxact:parameters	The root element to define a parameter. It requires the definition of attributes and

Element/Attribute	Parent Element	Content
(or ipxact:moduleParameter)	(or ipxact:moduleParameters)	children element for the proper description of a parameter
resolve	attribute of ipxact:parameter (or ipxact:moduleParameter)	Can take one of the values: "user", "immediate" or "generated". To specify that a parameter should be configured by the user of the IP, the value **"user"** should be used. This will also display the parameter in the properties dialog of an IP inside PathWave FPGA This attribute defaults to "immediate" if not defined
type	attribute of ipxact:parameter (or ipxact:moduleParameter)	Defines the datatype of the value. Possible values are: "int", "bit", "byte". For a complete list, please refer to IP-XACT 1685-2014
parameterId	attribute of ipxact:parameter (or ipxact:moduleParameter)	Defines a unique (in the context of the IP-XACT file) ID for this parameter. This ID should then be used in any expression required within the file
ipxact::name	ipxact:parameter (or ipxact:moduleParameter)	The name of the parameter
ipxact:value	ipxact:parameter (or ipxact:moduleParameter)	The default value (or expression) of the parameter

Component Parameters

Parameters defined as children of the elements path *component->parameters*. These can be used throughout the IP-XACT document to configure any aspect of the file (can be used in any field that accepts expressions as values, e.g. other parameter values, port ranges, port presence etc.)

```
<ipxact:component>
    :
    :
 <ipxact:parameters>
      <ipxact:parameter resolve="user" type="int"
parameterId="gen_input_length" >
         <ipxact:name>gen_input_length</ipxact:name>

 <ipxact:value>3*uuid_5e192450_89f2_48a9_8906_ee47dbbe0b15</ipxact:value>
      </ipxact:parameter>
      <ipxact:parameter type="int"
parameterId="uuid_f4a7c3f8_a1b3_496a_9730_17d721278396" >
         <ipxact:name>output_length</ipxact:name>
         <ipxact:value>2*gen_input_length</ipxact:value>
      </ipxact:parameter>
      <ipxact:parameter resolve="user" type="int"
parameterId="uuid_5e192450_89f2_48a9_8906_ee47dbbe0b15" >
         <ipxact:name>supersample</ipxact:name>
         <ipxact:value>1</ipxact:value>
```

```
      </ipxact:parameter>
 </ipxact:parameters>
    :
    :
</ipxact:component>
```

Notes:

- The tag *resolve="user"* indicates that these parameters are ones that the user can change when instantiating the IP block. If the parameter should always be calculated from other values or remain fixed, the tag *resolve="immediate"* should be used. In that case the user will not be given the option of modifying the value of the parameter.
- The *parameterId* is the one used inside an expression (**not** the ipxact:name), in which a parameter participates (see ipxact:value of parameter gen_input_length). However, if the *ipxact:name* of the parameter is unique throughout the document, it can also be used as *parameterId*. This way it is easier to construct expressions using parameters (see ipxact:value of parameter *output_length*)
- The value of *output_length* parameter **shall not** be modifiable **directly** by user input (as it does not contain the attribute *resolve* set to *"user"*), rather, **indirectly**, through the *input_length* parameter, as its expression implies (i.e. *2*gen_input_length*)
- The value of gen_input_length parameter is defined as user modifiable. That means that the expression **shall not** play any role, other than defining the default value. Therefore, if a user selects a value of "10" for this parameter, and a value of "5" for the parameter *supersample*, the final value of gen_input_length will be "10" and **not** "15" (*3*supersample*)

Module Parameters

Parameters defined as children of the elements path *component->model->instantiations->componentInstantiation->moduleParameters*. These are more specific to a Module Definition. Represent the *generics* of a VHDL *entity*, or the *parameters* of a Verilog *module.*

Code Block 4 Example Module Parameters Definition

```
<ipxact:component>
    :
    :
    <ipxact:model>
        <ipxact:instantiations>
            <ipxact:componentInstantiation>
                <ipxact:name>flat_vhdl_component</ipxact:name>
            <ipxact:language>vhdl</ipxact:language>
            <ipxact:moduleName>parameterizedIp</ipxact:moduleName>
            <ipxact:moduleParameters>
               <ipxact:moduleParameter type="int"
parameterId="input_length" resolve="user">
                    <ipxact:name>input_length</ipxact:name>
                    <ipxact:value>3*supersample</ipxact:value>
                 </ipxact:moduleParameter>
               <ipxact:moduleParameter type="int"
```

```
parameterId="output_length">
                    <ipxact:name>output_length</ipxact:name>
                    <ipxact:value>2*input_length</ipxact:value>
                  </ipxact:moduleParameter>
               <ipxact:moduleParameter type="int"
parameterId="supersample" resolve="user">
                    <ipxact:name>supersample</ipxact:name>

 <ipxact:value>uuid_5e192450_89f2_48a9_8906_ee47dbbe0b15</ipxact:value>
                  </ipxact:moduleParameter>
            </ipxact:moduleParameters>
       </ipxact:componentInstantiation>
    </ipxact:instantiations>
    :
    :
 </ipxact:model>
    :
    :
</ipxact:component>
```

Notes:

- The guides for creating component parameters also apply to the module parameters.
- The value of the *supersample* parameter depends on a parameter defined elsewhere in the document (*uuid_5e192450_89f2_48a9_8906_ee47dbbe0b15* is the *parameterId* defined for the parameter supersample, defined in the previous example and can exist in the same document)

Example: Parameterized Port Sizing

IP-XACT parameters can be used to define the bounds (sizes) of the IP module's ports. These expressions may be solely the parameterId of an ipxact:moduleParameter or may be more complicated expressions as shown in this example:

```
<ipxact:port>
    <ipxact:name>Din_vector</ipxact:name>
    <ipxact:wire>
        <ipxact:direction>in</ipxact:direction>
        <ipxact:vectors>
            <ipxact:vector>
                <ipxact:left>input_length*supersample-1</ipxact:left>
                <ipxact:right>0</ipxact:right>
            </ipxact:vector>
        </ipxact:vectors>
        <ipxact:wireTypeDefs>
            <ipxact:wireTypeDef>
                <ipxact:typeName>std_logic_vector</ipxact:typeName>
                <ipxact:typeDefinition></ipxact:typeDefinition>
            </ipxact:wireTypeDef>
        </ipxact:wireTypeDefs>
    </ipxact:wire>
</ipxact:port>
```

Note:

- **Only ipxact:moduleParameter parameters can be used in expressions defining port ranges**. This is because the actual expression will also be used during code generation and only the ipxact:moduleParameters are defined at that time
- Tools such as *Kactus2* can facilitate defining and evaluating expressions.

IP Restrictions

For IP to be used in PathWave FPGA, there will need to be a set of IP restrictions that specify which BSPs and FPGA device families the IP can be used with. This information will be used to determine which IP will show up in the IP catalog in the GUI for use in a design. Only the IP that will work with a given BSP and FPGA will show up for a design so that the user cannot place incompatible IP in a design.

An IP developer may specify in the IP-XACT which BSPs (eg. M3102A, M3202A), which FPGA vendors (eg. Xilinx), and which FPGA families (eg. Virtex, Kintex) are supported. If the IP can work for all families for a given FPGA vendor or all BSPs, then the family parameter or the bsp parameter does not need to be set.

IP Restrictions Format

The IP restrictions will be added to the IP-XACT file inside the 'ipxact:vendorExtensions' element of an 'ipxact:component'. The elements to be used are defined by Keysight and are as follows:

Element	Parent Element	Content
keysight:ipMetadata	ipxact:vendorExtensions (direct child of ipxact:component)	This is the root element of the Keysight Vendor Extensions for IP metadata
keysight:supportedHardware	keysight:ipMetadata	Contains information about the hardware to which this IP is supported
keysight:supportedBoards	keysight:supportedHardware	Contains a list of Vendor-Boards pairs of supported boards. **If this element is not specified, all boards are supported**
keysight:vendorBoards	keysight:supportedBoards	A Vendor-Boards pair
keysight:vendor	keysight:vendorBoards	The name of the vendor. Should be equal to the internet domain of the vendor of the boards (e.g. keysight.com)
keysight:boards	keysight:vendorBoards	Contains a list of board names that are supported
keysight:board	keysight:boards	The name of the board where this IP can be used
keysight:supportedParts	keysight:supportedHardware	Contains a list of Vendor-Parts pairs of supported FPGA parts. **If this element is not specified, all FPGA parts are supported**
keysight:vendorParts	keysight:supportedParts	A Vendor-Parts pair
keysight:vendor	keysight:vendorParts	Vendor's name. Should be equal to the internet domain of

Element	Parent Element	Content
		the vendor of the parts (e.g. keysight.com)
keysight:families	keysight:vendorParts	Contains a list of family names that are supported
keysight:family	keysight:families	The name of the family as this is defined by the part number (e.g. 'xc7k' should be used if the supported family is 'Kintex-7')

To use any of the Keysight defined elements inside an IP-XACT file, you need to specify the 'keysight' namespace: "xmlns:keysight="http://www.keysight.com"" in the xml root element (i.e. ipxact:component)

IP Categorization

In addition to defining the library in which the IP belongs, it is possible to define a subcategory for an IP. To achieve that, PathWave FPGA has defined some extension elements for IP-XACT.

The IP restrictions will be added to the IP-XACT file inside the 'ipxact:vendorExtensions' element of an 'ipxact:component'. The elements to be used are defined by Keysight and are as follows:

Element	Parent Element	Content
keysight:ipMetadata	ipxact:vendorExtensions (direct child of ipxact:component)	This is the root element of the Keysight Vendor Extensions for IP metadata
keysight:categories	keysight:ipMetadata	A list of categories. Currently, only one category can be specified
keysight:category	keysight:categories	The name of the category that this IP belongs into

To use any of the Keysight defined elements inside an IP-XACT file, you need to specify the 'keysight' namespace: "xmlns:keysight="http://www.keysight.com"" in the xml root element (i.e. ipxact:component)

IP Naming Collisions

PathWave FPGA does not accept IP with the same name to be loaded at the same time in a project. PathWave FPGA uses the concept of VLNV for identifying IP and reporting naming collisions. VLNV stands for Vendor-Library-Name-Version and is a concept introduced by IP-XACT. The VLNV of an IP is defined in the first four fields of an IP-XACT component (see IP-XACT definition)

For more information on naming collisions and how to resolve them, please read here.

For the case of an IP developer, this might happen as multiple versions of the same IP might be created in the development phase. Even though the case of multiple IPs with the same VLNV but different contents is detected by PathWave FPGA, it is recommended to update the version field of the IP-XACT file for every change applied to the file. This will provide better issue reporting and easier resolution.

An Example IP-XACT File

Code Block 5 Sample IP-XACT file

```
<?xml version="1.0" encoding="UTF-8"?>
<ipxact:component xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014"
xmlns:keysight="http://www.keysight.com"
xsi:schemaLocation="http://www.accellera.org/XMLSchema/IPXACT/1685-2014
http://www.accellera.org/XMLSchema/IPXACT/1685-2014/index.xsd">
  <ipxact:vendor>keysight.com</ipxact:vendor>
  <ipxact:library>myCustomLibrary</ipxact:library>
  <ipxact:name>SampleIp</ipxact:name>
  <ipxact:version>1.0</ipxact:version>
  <ipxact:busInterfaces>
    <ipxact:busInterface>
      <ipxact:name>clkSignal</ipxact:name>
      <ipxact:busType vendor="keysight.com" library="interfaces"
name="clock" version="1.0"/>
      <ipxact:abstractionTypes>
        <ipxact:abstractionType>
          <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="clock.absDef" version="1.0"/>
          <ipxact:portMaps>
            <ipxact:portMap>
              <ipxact:logicalPort>
                <ipxact:name>clk</ipxact:name>
              </ipxact:logicalPort>
              <ipxact:physicalPort>
                <ipxact:name>clk</ipxact:name>
              </ipxact:physicalPort>
            </ipxact:portMap>
          </ipxact:portMaps>
        </ipxact:abstractionType>
      </ipxact:abstractionTypes>
      <ipxact:slave/>
      <ipxact:parameters>
        <ipxact:parameter
parameterId="uuid_4e5d34f4_ff5d_4244_92b4_c0d0ec78d043">
          <ipxact:name>ASSOCIATED_BUSIF</ipxact:name>
          <ipxact:value>myAxiStreamMaster:myAxiStreamSlave</ipxact:value>
        </ipxact:parameter>
        <ipxact:parameter
parameterId="uuid_c127b078_eb51_42f4_aaf8_58e93ad84b21">
          <ipxact:name>ASSOCIATED_RESET</ipxact:name>
          <ipxact:value>Reset</ipxact:value>
        </ipxact:parameter>
      </ipxact:parameters>
    </ipxact:busInterface>
    <ipxact:busInterface>
      <ipxact:name>Reset</ipxact:name>
      <ipxact:busType vendor="keysight.com" library="interfaces"
name="nRst" version="1.0"/>
```

```
      <ipxact:abstractionTypes>
        <ipxact:abstractionType>
          <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="nRst.absDef" version="1.0"/>
          <ipxact:portMaps>
            <ipxact:portMap>
              <ipxact:logicalPort>
                <ipxact:name>nRst</ipxact:name>
              </ipxact:logicalPort>
              <ipxact:physicalPort>
                <ipxact:name>rstn</ipxact:name>
              </ipxact:physicalPort>
            </ipxact:portMap>
          </ipxact:portMaps>
        </ipxact:abstractionType>
      </ipxact:abstractionTypes>
      <ipxact:slave/>
    </ipxact:busInterface>
    <ipxact:busInterface>
      <ipxact:name>myAxiStreamSlave</ipxact:name>
      <ipxact:busType vendor="keysight.com" library="interfaces"
name="axis" version="1.0"/>
      <ipxact:abstractionTypes>
        <ipxact:abstractionType>
          <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="axis.absDef" version="1.0"/>
          <ipxact:portMaps>
            <ipxact:portMap>
              <ipxact:logicalPort>
                <ipxact:name>tvalid</ipxact:name>
              </ipxact:logicalPort>
              <ipxact:physicalPort>
                <ipxact:name>my_stream_valid_in</ipxact:name>
              </ipxact:physicalPort>
            </ipxact:portMap>
            <ipxact:portMap>
              <ipxact:logicalPort>
                <ipxact:name>tuser</ipxact:name>
              </ipxact:logicalPort>
              <ipxact:physicalPort>
                <ipxact:name>my_stream_user_in</ipxact:name>
              </ipxact:physicalPort>
            </ipxact:portMap>
            <ipxact:portMap>
              <ipxact:logicalPort>
                <ipxact:name>tdata</ipxact:name>
              </ipxact:logicalPort>
              <ipxact:physicalPort>
                <ipxact:name>my_stream_data_in</ipxact:name>
              </ipxact:physicalPort>
            </ipxact:portMap>
          </ipxact:portMaps>
        </ipxact:abstractionType>
      </ipxact:abstractionTypes>
      <ipxact:slave/>
    </ipxact:busInterface>
    <ipxact:busInterface>
      <ipxact:name>myAxiStreamMaster</ipxact:name>
      <ipxact:busType vendor="keysight.com" library="interfaces"
name="axis" version="1.0"/>
      <ipxact:abstractionTypes>
        <ipxact:abstractionType>
          <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="axis.absDef" version="1.0"/>
          <ipxact:portMaps>
            <ipxact:portMap>
```

```
            <ipxact:logicalPort>
              <ipxact:name>tvalid</ipxact:name>
            </ipxact:logicalPort>
            <ipxact:physicalPort>
              <ipxact:name>my_stream_valid_out</ipxact:name>
            </ipxact:physicalPort>
          </ipxact:portMap>
          <ipxact:portMap>
            <ipxact:logicalPort>
              <ipxact:name>tdata</ipxact:name>
            </ipxact:logicalPort>
            <ipxact:physicalPort>
              <ipxact:name>my_stream_data_out</ipxact:name>
            </ipxact:physicalPort>
          </ipxact:portMap>
        </ipxact:portMaps>
      </ipxact:abstractionType>
    </ipxact:abstractionTypes>
    <ipxact:master/>
  </ipxact:busInterface>
</ipxact:busInterfaces>
<ipxact:model>
  <ipxact:ports>
    <ipxact:port>
      <ipxact:name>clk</ipxact:name>
      <ipxact:wire>
        <ipxact:direction>in</ipxact:direction>
        <ipxact:wireTypeDefs>
          <ipxact:wireTypeDef>
            <ipxact:typeName>std_logic</ipxact:typeName>
            <ipxact:typeDefinition></ipxact:typeDefinition>
          </ipxact:wireTypeDef>
        </ipxact:wireTypeDefs>
      </ipxact:wire>
    </ipxact:port>
    <ipxact:port>
      <ipxact:name>rstn</ipxact:name>
      <ipxact:wire>
        <ipxact:direction>in</ipxact:direction>
        <ipxact:wireTypeDefs>
          <ipxact:wireTypeDef>
            <ipxact:typeName>std_logic</ipxact:typeName>
            <ipxact:typeDefinition></ipxact:typeDefinition>
          </ipxact:wireTypeDef>
        </ipxact:wireTypeDefs>
      </ipxact:wire>
    </ipxact:port>
    <ipxact:port>
      <ipxact:name>my_stream_valid_in</ipxact:name>
      <ipxact:wire>
        <ipxact:direction>in</ipxact:direction>
        <ipxact:wireTypeDefs>
          <ipxact:wireTypeDef>
            <ipxact:typeName>std_logic</ipxact:typeName>
            <ipxact:typeDefinition></ipxact:typeDefinition>
          </ipxact:wireTypeDef>
        </ipxact:wireTypeDefs>
      </ipxact:wire>
    </ipxact:port>
    <ipxact:port>
      <ipxact:name>my_stream_data_in</ipxact:name>
      <ipxact:wire>
        <ipxact:direction>in</ipxact:direction>
        <ipxact:vectors>
          <ipxact:vector>
            <ipxact:left>79</ipxact:left>
```

```
            <ipxact:right>0</ipxact:right>
          </ipxact:vector>
        </ipxact:vectors>
        <ipxact:wireTypeDefs>
          <ipxact:wireTypeDef>
            <ipxact:typeName>std_logic_vector</ipxact:typeName>
            <ipxact:typeDefinition></ipxact:typeDefinition>
          </ipxact:wireTypeDef>
        </ipxact:wireTypeDefs>
      </ipxact:wire>
    </ipxact:port>
    <ipxact:port>
      <ipxact:name>my_stream_user_in</ipxact:name>
      <ipxact:wire>
        <ipxact:direction>in</ipxact:direction>
        <ipxact:vectors>
          <ipxact:vector>
            <ipxact:left>0</ipxact:left>
            <ipxact:right>0</ipxact:right>
          </ipxact:vector>
        </ipxact:vectors>
        <ipxact:wireTypeDefs>
          <ipxact:wireTypeDef>
            <ipxact:typeName>std_logic_vector</ipxact:typeName>
            <ipxact:typeDefinition></ipxact:typeDefinition>
          </ipxact:wireTypeDef>
        </ipxact:wireTypeDefs>
      </ipxact:wire>
    </ipxact:port>
    <ipxact:port>
      <ipxact:name>my_stream_valid_out</ipxact:name>
      <ipxact:wire>
        <ipxact:direction>out</ipxact:direction>
        <ipxact:wireTypeDefs>
          <ipxact:wireTypeDef>
            <ipxact:typeName>std_logic</ipxact:typeName>
            <ipxact:typeDefinition></ipxact:typeDefinition>
          </ipxact:wireTypeDef>
        </ipxact:wireTypeDefs>
      </ipxact:wire>
    </ipxact:port>
    <ipxact:port>
      <ipxact:name>my_stream_data_out</ipxact:name>
      <ipxact:wire>
        <ipxact:direction>out</ipxact:direction>
        <ipxact:vectors>
          <ipxact:vector>
            <ipxact:left>79</ipxact:left>
            <ipxact:right>0</ipxact:right>
          </ipxact:vector>
        </ipxact:vectors>
        <ipxact:wireTypeDefs>
          <ipxact:wireTypeDef>
            <ipxact:typeName>std_logic_vector</ipxact:typeName>
            <ipxact:typeDefinition></ipxact:typeDefinition>
          </ipxact:wireTypeDef>
        </ipxact:wireTypeDefs>
      </ipxact:wire>
    </ipxact:port>
  </ipxact:ports>
</ipxact:model>
<ipxact:fileSets>
  <ipxact:fileSet>
    <ipxact:name>synthesis</ipxact:name>
    <ipxact:file>
      <ipxact:name>sampleIp.vhd</ipxact:name>
```

```
        <ipxact:fileType>vhdlSource</ipxact:fileType>
      </ipxact:file>
    </ipxact:fileSet>
  </ipxact:fileSets>
  <ipxact:description>This is a Sample IP. It contains two Stream
Interfaces and two system ports</ipxact:description>
  <ipxact:vendorExtensions>
    <keysight:ipMetadata>
      <keysight:supportedHardware>
        <keysight:supportedBoards>
          <keysight:vendorBoards>
            <keysight:vendor>keysight.com</keysight:vendor>
            <keysight:boards>
              <keysight:board>M3202A</keysight:board>
            </keysight:boards>
          </keysight:vendorBoards>
        </keysight:supportedBoards>
        <keysight:supportedParts>
          <keysight:vendorParts>
            <keysight:vendor>xilinx.com</keysight:vendor>
            <keysight:families>
              <keysight:family>xc7k</keysight:family>
            </keysight:families>
          </keysight:vendorParts>
        </keysight:supportedParts>
      </keysight:supportedHardware>
      <keysight:categories>
        <keysight:category>General</keysight:category>
      </keysight:categories>
    </keysight:ipMetadata>
  </ipxact:vendorExtensions>
</ipxact:component>
```

Keysight Standard Interfaces

Introduction

To facilitate connectivity between IP blocks and Sandbox interfaces, PathWave FPGA has standardized on a number of interfaces. IP blocks using these interfaces will be easier to interconnect and to connect to PathWave FPGA library blocks and sandbox interfaces.

Interface Descriptions

The following is a brief description of the standard interfaces PathWave FPGA supports. Note that this is only a brief description of each interface and is not meant to be a complete description. Some interfaces (e.g. the AXI family) include optional signals that can be included or omitted in particular implementations depending on the design requirements. This allows the user to tailor the complexity and size of the interface while maintaining compatibility.

1. clock: A free running clock. Data is both sampled and changed on the rising edge of a clock.
2. nRst: An active low reset signal.
3. AXIMM: the industry standard, AXI4-Memory Mapped high performance bus architecture.
    a. Includes address information.
    b. Supports data widths: 8, 16, 32, 64, 128, 256, 512, 1024 bits.
    c. Supports burst (high performance) transfers.
    d. Supports bi-directional flow control.
4. AXILite: the AXI4-Lite bus, a lightweight version of AXIMM for simpler interfaces that don't require the performance/features of full blown AXI4.
    a. Limited data width: 32 (preferred) or 64 (if needed).
    b. Only single transactions supported - no data bursting.
    c. Supports bi-directional flow control.
5. AXIS: the AXI4-Streaming interface is for streaming arbitrarily long sequences of data.
    a. Point-to-point streams - this interface does not include address data, though optional TID, and TDEST signals allow some routing (addressing) information.
    b. Data width is any multiple of 8 bits. Unlike AXIMM and AXILite, AXIS can support, for example, 24 bit data. The standard allows 0 bit data (TDATA is optional). An AXIS interface without data just has the control signals.
    c. Supports optional TUSER data signals. These are extra signals that are logically attached to data samples that could be used to include auxiliary data such as triggers or data marks or timing information.
    d. Supports merging/packing multiple data items into wider stream.
    e. Supports bi-directional flow control.
6. PC-MEM: a very light weight Keysight proprietary interface.
    a. Can be bi-directional.
    b. Includes addressing.
    c. Does not include back-pressure - all transactions take place in one clock cycle and can not be held off.
    d. Has deterministic timing.
    e. Used for HVI register access. Please see the Keysight M3601 documentation for more information on HVI.
7. vector: a multi-bit vector of signals without any signaling protocol. This might be used to connect a control register to an IP block.

8. wire: a single bit signal. This might be used for a trigger signal.

Signal Types

There are a number of different types of signals used in a typical design. These can roughly be categorized into control signals (typically used to setup, control, and monitor a measurement), data flow signals (the data being processed - this could be a continuous stream of data or one or more blocks of data), and a miscellaneous category containing things like triggers, timestamps, etc.

The following are the various types of signals that PathWave FPGA supports:

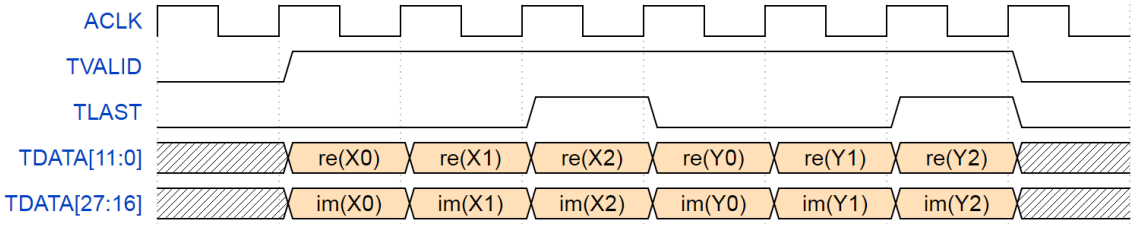
1. Control Bus Slaves. Typically these would be register control/status blocks where the driver could read and write status and control data.
2. Control Bus Master. This is for the case where the user IP wants to communicate with external devices via the PCIe (or other host control) bus, e.g. write to other modules to control multi-module measurements.
3. Continuous Streaming Data. This is an arbitrarily long stream of continuous data, e.g. from an ADC. Since the data may not be one sample per clock, flow control is required. Alongside the data, there may optionally be some amount of sideband data. This is auxiliary data that flows along with the main signal data. It could include triggers or marker info or be used to timestamp data.
4. Block Mode Stream Data. This would be an arbitrarily long stream of discontinuous blocks of data. Each block may represent the result of some measurement or calculation, e.g. the output of an FFT. To properly interpret this data, the boundaries of each block would need to be delineated.
5. Memory Read / Write Data. Typically the FPGA will have access to off chip memory. There needs to be a way for the user IP to read and write to this memory. This interface will need to include both address and data flow, and probably needs to support burst transfers for efficiency.
6. Supersampled Data. This is a variation of #3 and #4 above where more than one sample per clock needs to be transferred.
7. HVI. HVI needs an efficient, time deterministic mechanism to access control register.
8. Clock. One or more clocks. Signals change on and are sampled on the rising edge of clock.
9. Reset. One or more active low reset signals.
10. Trigger. One or more rising edge or active high trigger signals.

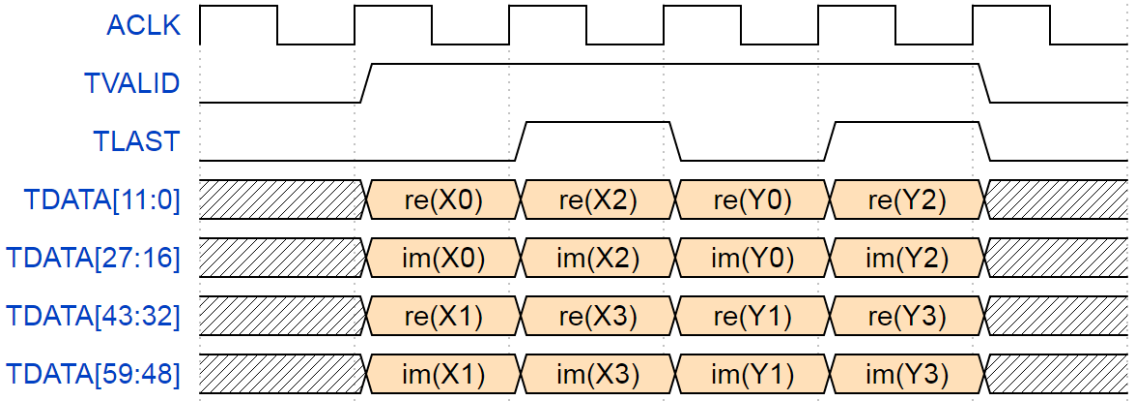
Data Types

Most of the data that PathWave FPGA will be processing is likely to be fixed point (scaled ints) of varying bit widths. To facilitate interconnection of IP, limit the amount of data width conversion, and allow the use of standard interfaces, PathWave FPGA standardizes on data widths that are an integral number of bytes (i.e. multiples of 8 bits). Data that is natively a different size should be padded up to the next multiple of 8 bits by padding MSBs. Unsigned quantities are zero-extended, and signed quantities are sign extended. Thus a 12 bit unsigned number would place those 12 bits as the 12 LSBs of the interface with the 4 MSBs being zero. So if the data was X[11:0], the interface used would be TDATA[15:0] = {4'b0000,X[11:0]}.

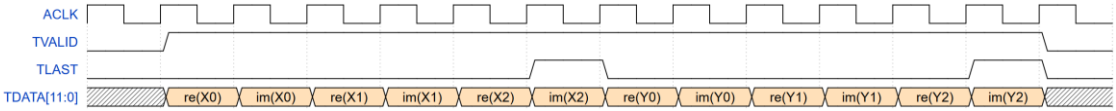
The preferred format for floating point numbers in PathWave FPGA will be IEEE-754 compliant. The two supported (preferred) sizes will be binary16 (16 bits with 11 bit fraction and 5 bit exponent) and binary32 (32 bits with 24 bit fraction and 8 bit exponent). Note that the number of fractional bits includes the implied leading "1" bit. The number of physical mantissa bits is one less than the number of fractional bits, and there is also sign bit. Physically, the binary32 format would have 1 sign bit, 8 exponent bits, and 23 mantissa bits.

It is not uncommon to process complex data (that is, data consisting of a real and an imaginary component). If complex data is being sent over a single stream, the real and imaginary parts will be sent in parallel over a wider stream with the real part will go in the least significant word. For Serial data, the real part will come first (earlier in time).

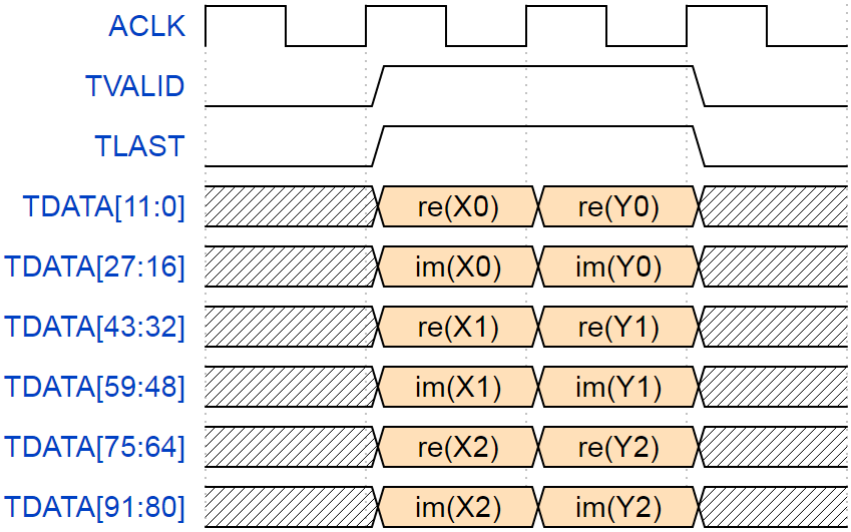




Above are examples of parallel complex data (one sample per clock and two samples per clock). Below is an example of serial complex data.



For performance reasons (and the limited clock rate available in FPGAs), it is sometimes desired to transfer more than one sample per clock. This is called *supersampled* data. In this case, each sample (or component of the sample for complex data) is first extended to an integral number of bytes, and then these are packed together with the earlier in time samples occupying the lesser significant position:



Data Packing/Extending

When connecting two blocks with different data widths, there are two different ways of converting the signals. The AXI standard views data as a stream of bytes without explicit meaning. Going from a narrow to a wider interface will cause the bytes to be packed. For example, going from a 16 bit interface to a 32 bit interface will pack two 16 bit words into each 32 bit word. Likewise going from a wide to a narrow interface will retain all the data bytes with the output running at a higher rate than the input. This is desired behavior when interfacing to a memory, for example.

The other situation is when the underlying bit widths of the data changes, for example when interfacing a filter that uses 16 bit data to a filter using 24 bit data. When increasing the width (e.g. 16 bit source feeding a 24 bit sink) the data should be sign extended per PathWave FPGA's policy of right justifying fixed point data.

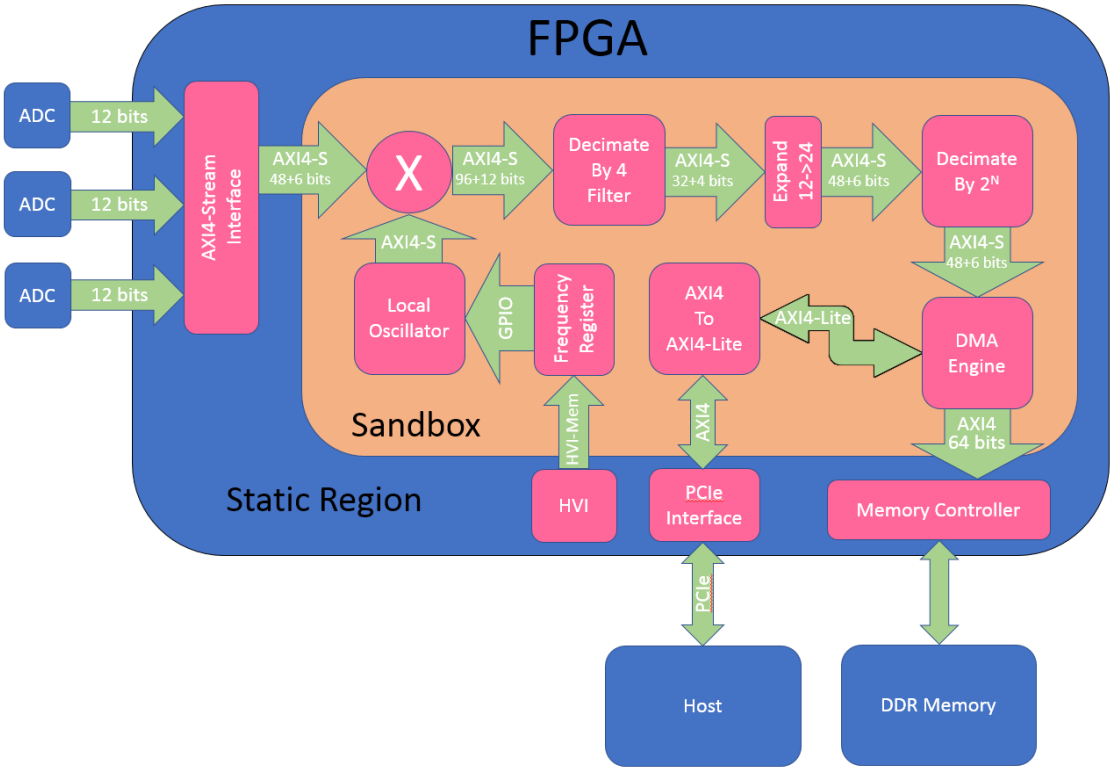
Polarity

The control signals for the AXI buses are generally active high. The exception is the nRST signal which is active low. PathWave FPGA uses an active low nRST signal. The remaining control signals should be active high. Further, PathWave FPGA should sample signals and change signals on the rising edge of CLK.

Signal Interfaces

Signal Type	Interface	Discussion
Clock	clock	One or more free running clocks. Signals change on and are sampled on the rising edge of clock.
Reset	nRst	One or more active low reset signals.
Control Bus Slaves	AXIMM AXILite	Most Control Bus Slaves can probably use the simpler AXILite interface. A simple block of registers can easily decode an AXILite interface with minimal logic. If higher performance of burst access is desired, then the higher capabilities of the full AXIMM bus could be used.
Control Bus Masters	AXIMM AXILite	These interfaces are full featured enough to meet the needs of IP that needs to instigate access to addressable memory/devices.
Continuous Streaming Data	AXIS	This interface supports the flow control and auxiliary data needs of continuous data transfers.
Block Mode Streaming Data	AXIS	This interface includes the TLAST signal that can be used to break the stream into arbitrary sized packets.
Memory Read/Write Data	AXIMM,AXILite, AXI4-Streaming	Memory, particularly off-chip memory, is generally used for storing larger amounts of data which often require high throughput accesses. If the user IP needs random access to the memory, then AXI4 is probably the better fit. If the memory is going to be used as a source or sink of streaming data, using a DMA engine in the static region, then an AXI4-Streaming interface would be a better fit.
Supersampled Streaming Data	AXI4-Streaming	As discussed above, if supersampled or complex data needs to be used, it will first be extended to an integral number of bytes and then packed into a wider AXIS interface.
PC-MEM	PC-MEM	Some addressable interfaces, such as HVI,have distinct, deterministic timing performance requirements. For very simple designs, this provides an ultra-lightweight, addressable interface.

Example Usage



Discussion of Example

This simplified example shows how these interfaces might be utilized.

In the above example, ADCs generate three parallel 12 bit samples per clock. In the static region these samples are converted to an AXIS bus as follows. Each sample is converted from 12 to 16 bits by sign extension. The resulting six bytes are concatenated together to form a 48 bit wide streaming data bus. One bit per byte of User data is added (six bits total) to contain trigger information. Note that these are more bits than necessary, but for compliance with the specification recommendations the extra (unneeded) bits are included.

The three real samples per clock are mixed with the output of a local oscillator to form three complex samples (96 bits total). The user data (still one bit per byte) is now 12 bits wide. Note that even though the interface into and out of the mixer is 16 bit data, since the user knows the data is only 12 bits wide, the internal logic of the multipliers in the mixer need only operate on 12 bits of data (ignoring the 4 extension bits).

After decimating by four, the data rate has been reduced to one complex sample per clock (actually 3/4 sample per clock - thus handshaking is needed) with the real and imaginary halves each using 16 bits. For increased dynamic range, the Decimate by 2^N block operates on 24 bit data rather than 12 bit. An expander widens the bus to 24 bit data (time two because it is complex). Note that the AXIS bus need not be a power of 2. It only has to be an integer number of bytes.

The output of the Decimate by 2^N block flows into a DMA Engine. This is designed to FIFO up the data and burst data via an AXIMM bus to the memory controller in the static region that will interface to the external DDR memory.

The Host controls the DMA Engine via the PCIe interface. The static region contains the PCIe interface and passes an AXIMM bus into the Sandbox. Since the registers controlling the DMA Engine are simple, there is no need for the DMA Engine to implement a full blown AXIMM interface. Instead, the AXIMM bus from the PCIe interface is converted to the simpler AXILite bus which feeds the registers in the DMA Engine.

For allowing synchronous measurements with other modules, the Frequency Register is controlled via time deterministic PC-Mem bus. The output of the Frequency Register is a plain Vector without control signals or handshake. This output controls the frequency of the Local Oscillator the output of which feeds the mixer.

Associated Files

AXI Reference Guide

Tutorials

- Import HDL with collapsible interfaces using IP-XACT
- Import HDL with parameterized bus widths using IP-XACT
- Import Vivado High-Level Synthesis (HLS) generated HDL with parameterized bus widths using IP-XACT

Import HDL with collapsible interfaces using IP-XACT

IP-XACT or IEEE 1685-2014 is an XML specification for describing (among other things) the interfaces used by an IP block in an FPGA. This tutorial describes the creation of an IP-XACT file for a simple IP block written in VHDL.

While the IP-XACT file is text and can be manually created in any text editor, it is simpler and easier to use an IP-XACT editor such as Kactus2 (available at http://funbase.cs.tut.fi/). PathWave FPGA recommends using version 3.5 or later.

The HDL IP block has physical ports which are the input and output signals for the IP block. One or more ports can be combined into logical interfaces which describe how the signals interact and connect with other signals. An interface may consist of a single port or even a signal wire. An example of this is a clock interface. Other interfaces, such as the AXI-MM interface, may have dozens of potential ports. By describing which ports constitute a particular interface and which role each port has, the IP-XACT description eases connecting interfaces together. An AXI Master can connect to an AXI Slave with only one connection even though a considerable number of individual ports will be connected in the hardware.

This tutorial will create the IP-XACT for the following simple block:

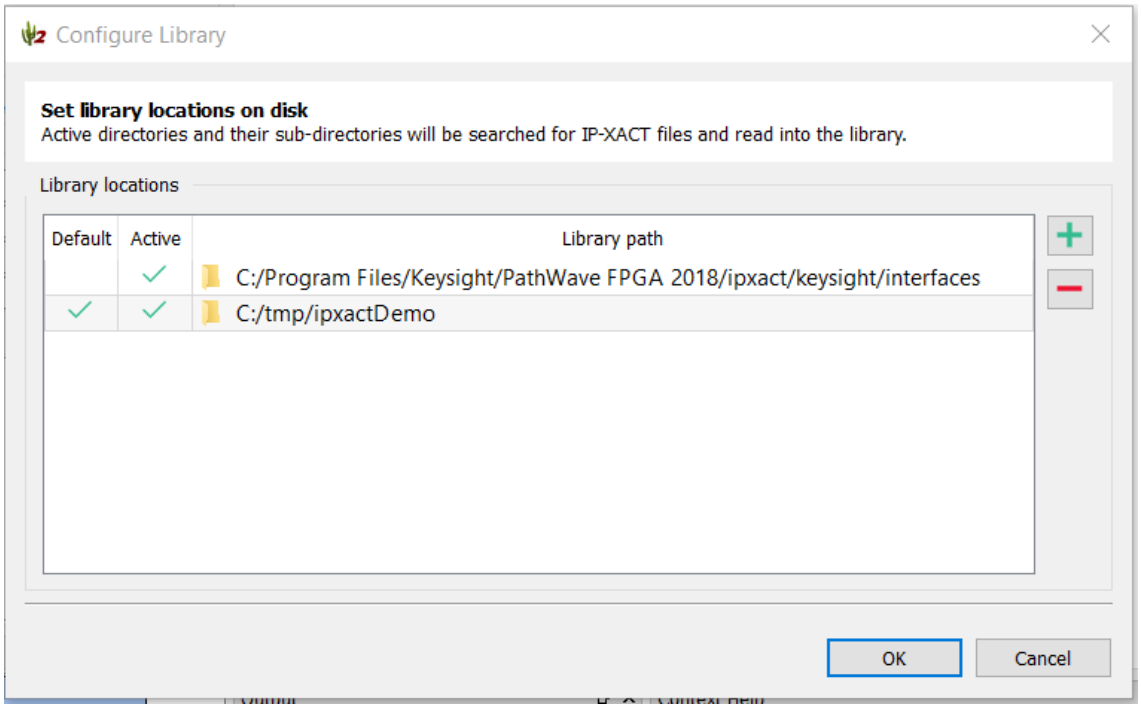
Code Block 6 incr1.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;


entity incr1 is

  port (
    clk   : in  std_logic;
    nrst  : in  std_logic;              -- Active low reset
    incr  : in  std_logic_vector(7 downto 0);
    count_tdata : out std_logic_vector(7 downto 0);
    count_tvalid : out std_logic
    );

end incr1;

architecture Behavioral of incr1 is
  signal count : std_logic_vector(7 downto 0);
begin  -- Behavioral
  count_tdata <= count;
  count_tvalid <= '1' when (incr /= 0) else '0';
  process(clk)
    begin
      if (nrst = '0') then
        count <= (others => '0');
      else
        count <= count + incr;
      end if;
    end process;
```

```
end Behavioral;
```

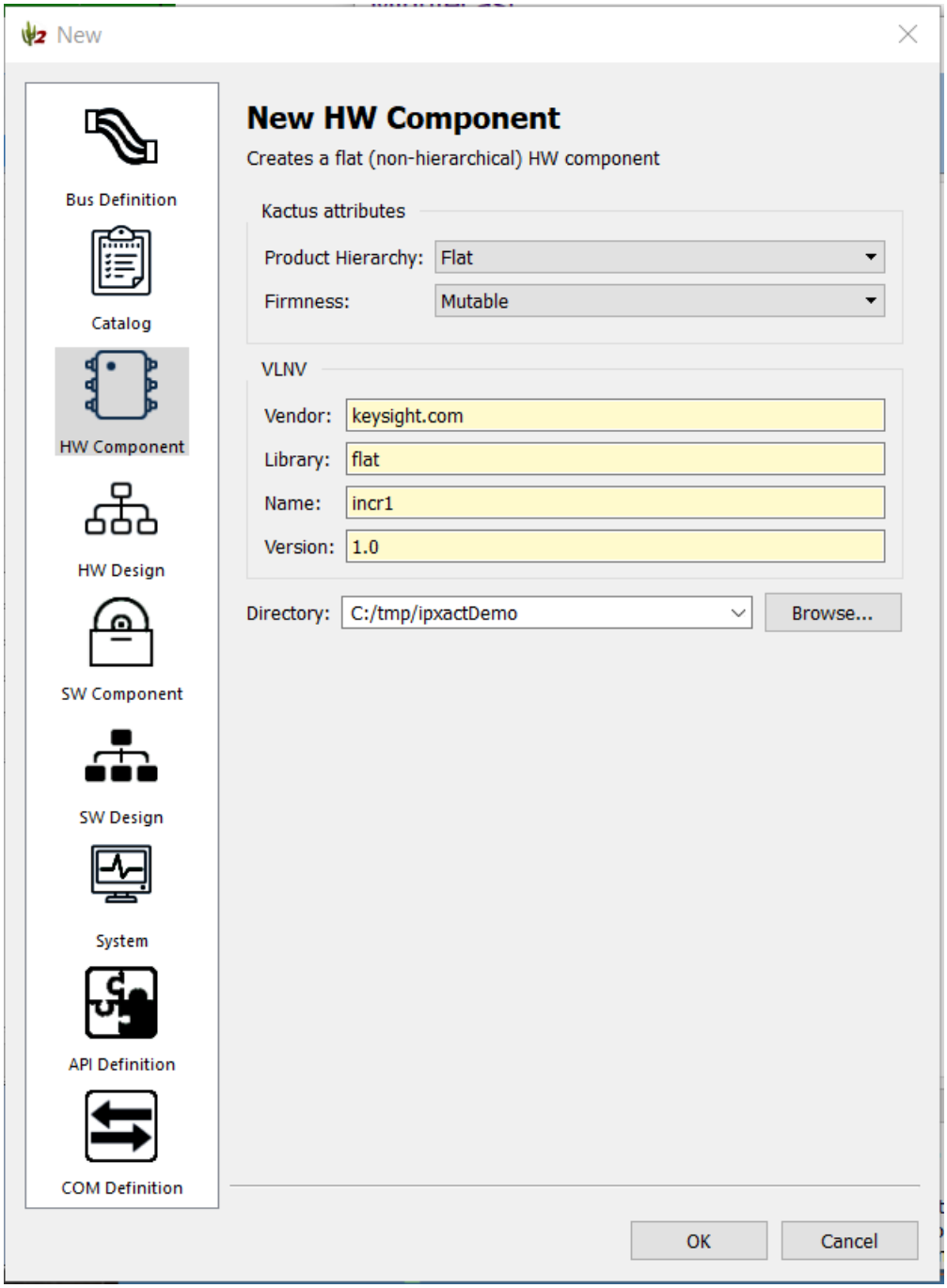
This block will increment an internal counter based on its *incr* input and output the counter value on an AXI-streaming *count* interface. It also has a clock input and an active low nrst input. This HDL file is stored under c:/tmp/ipxactDemo/src/incr1.vhd.

To create IP-XACT for this module, first start Kactus2. Click the **Configure Library** button to set up the libraries. Make sure that the PathWave FPGA interfaces folder as well as the folder for your IP block are both in the library path:

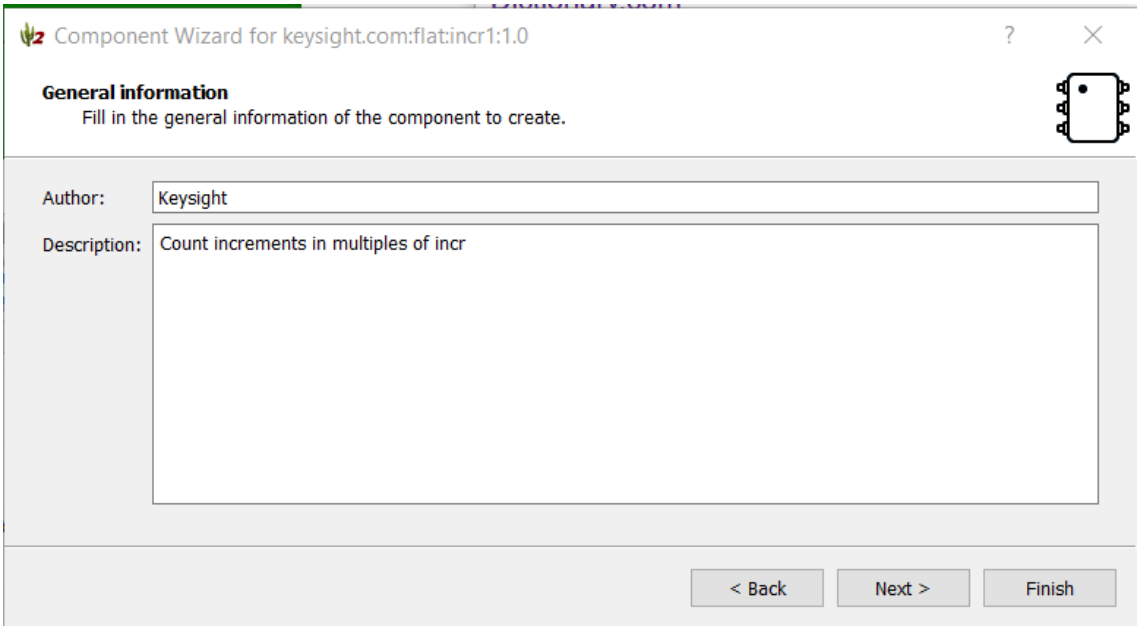


To create the IP-XACT, click the **New** button and select **HW Component.**

In Kactus2, required fields are shown in light yellow while optional fields are shown in white. Enter the **Vendor**, **Library**, **Name**, and **Version** for your IP block. Then click **Browse...** and navigate to the directory with the IP block. In this case, it will be c:/tmp/ipxactDemo. This is where the resulting IP-XACT file will be created:



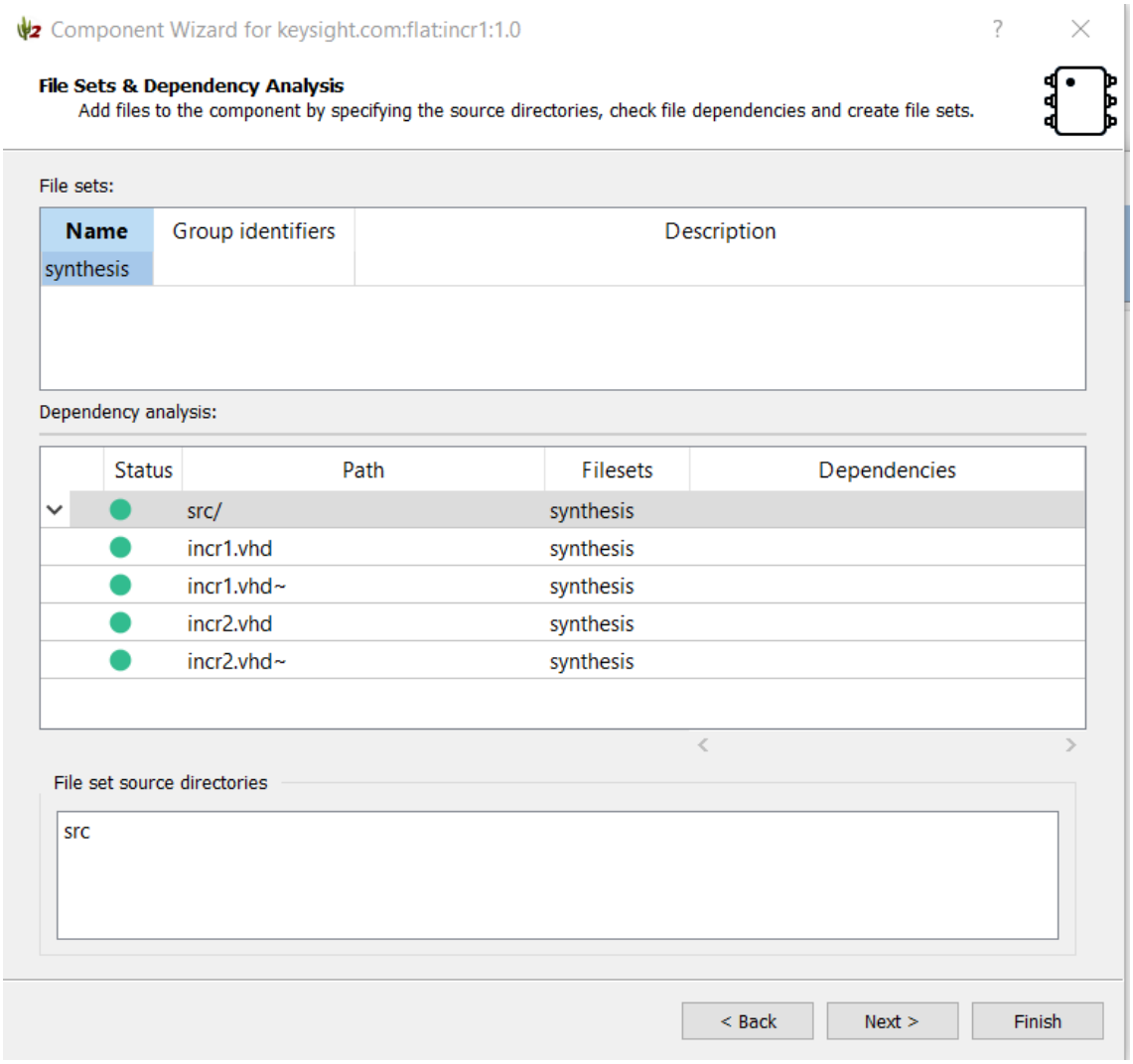
Click **OK** and the Component Wizard is started. Click **Next** to get to the General Information screen, and enter the Author and Description (which are optional):



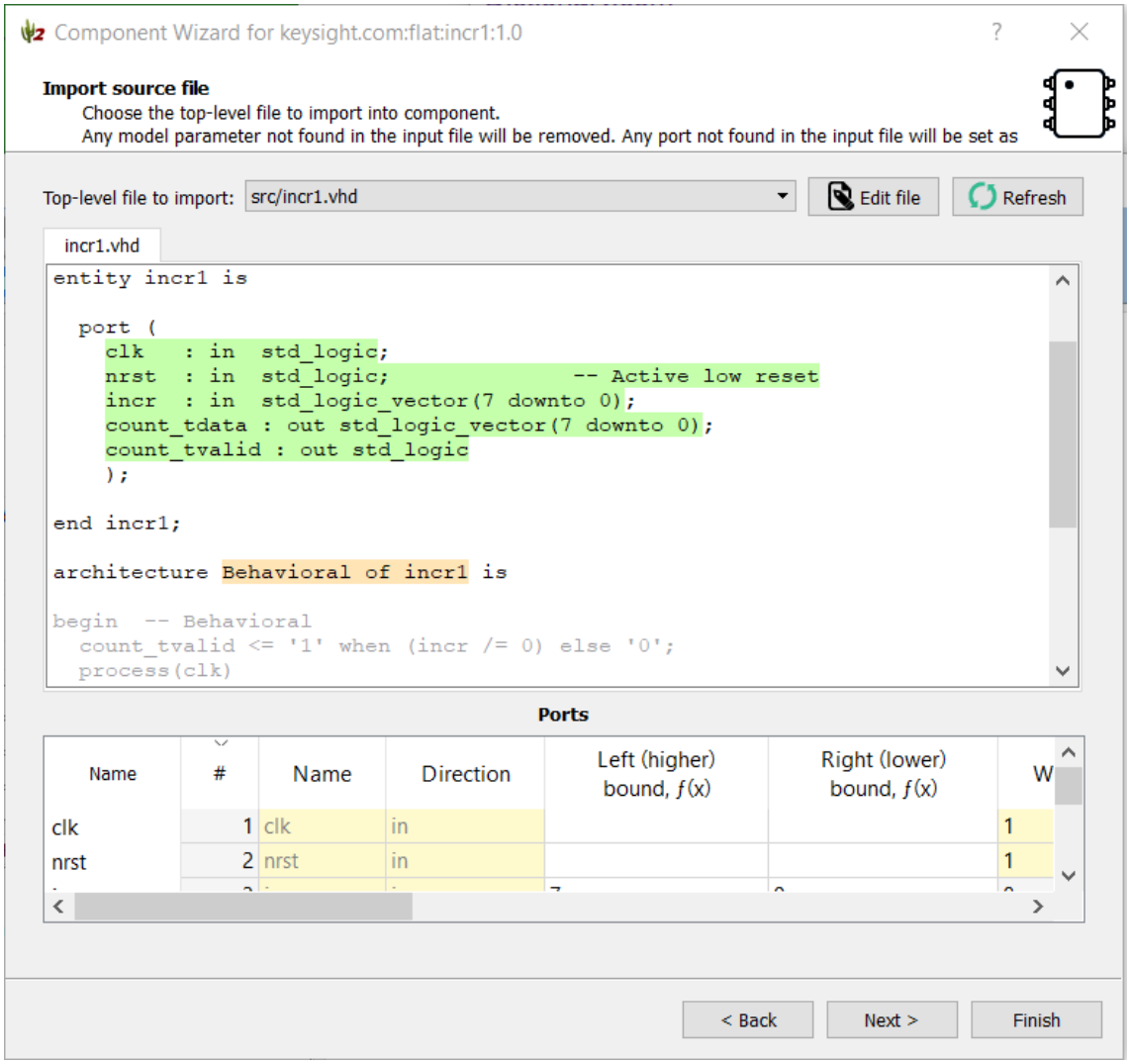
At this point, one could click **Finish** and proceed to enter the IP port information manually, but it is much more convenient to have Kactus2 read the source file and fill in the information automatically.
To do this, the source file folder needs to be set. Click **Next** to get to the File Sets & Dependency Analysis screen and double click in the **File set source directories** box to bring up the selection box.

Select the **src** directory and click **Select Folder.**

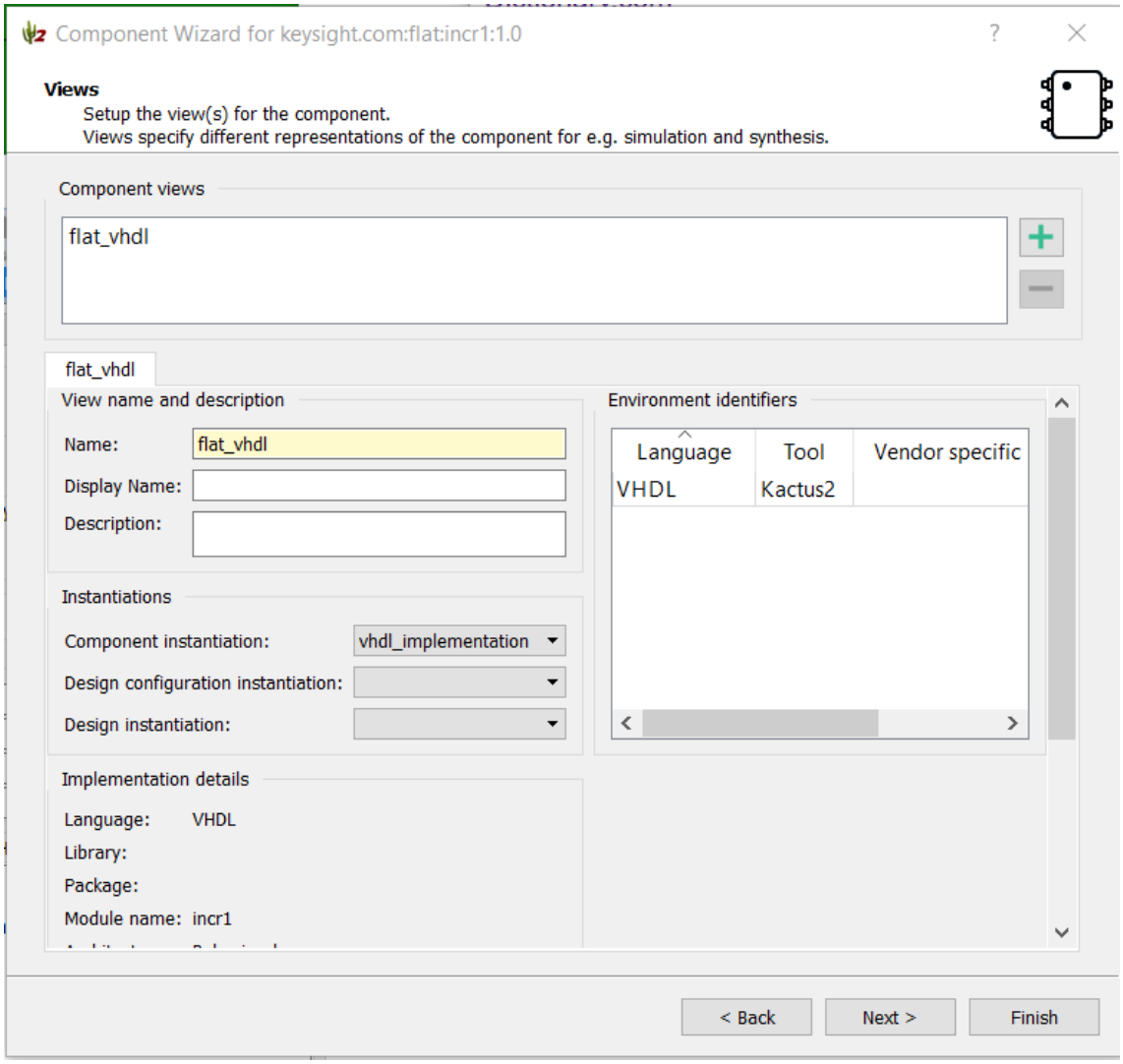
PathWave FPGA uses the **synthesis** fileset for containing the files needed for synthesis. Double click on the **File sets** / **Name** entry and change it to "synthesis":



Click **Next** to advance to the **Import source file** page. This is where the top level source file is specified. Using the pulldown menu for **Top-level file to import:**, select the incr1.vhd file:

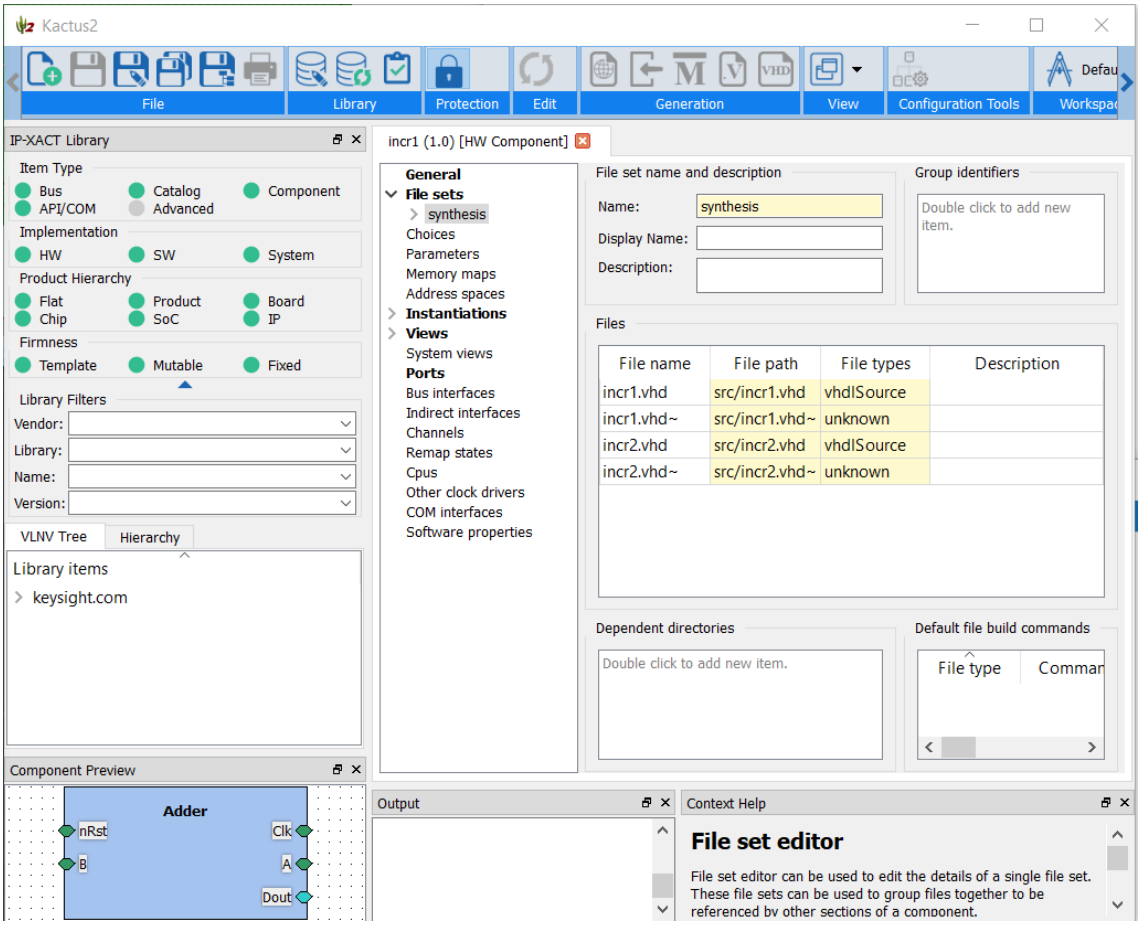


Note that the source file is shown in the middle pane with the detected ports in the lower pane. Click **Next** to advance to the **Views** page:

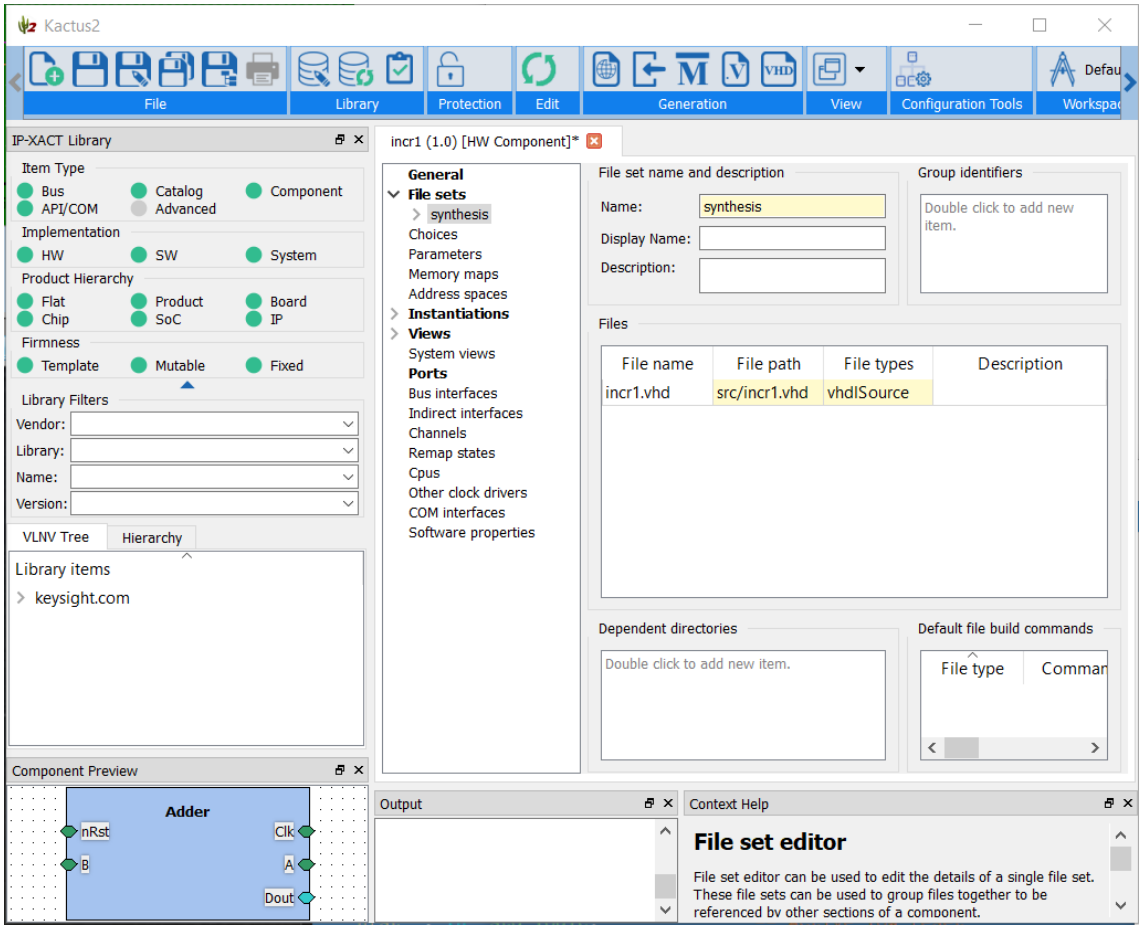


Click **Finish** to complete the Component Wizard.

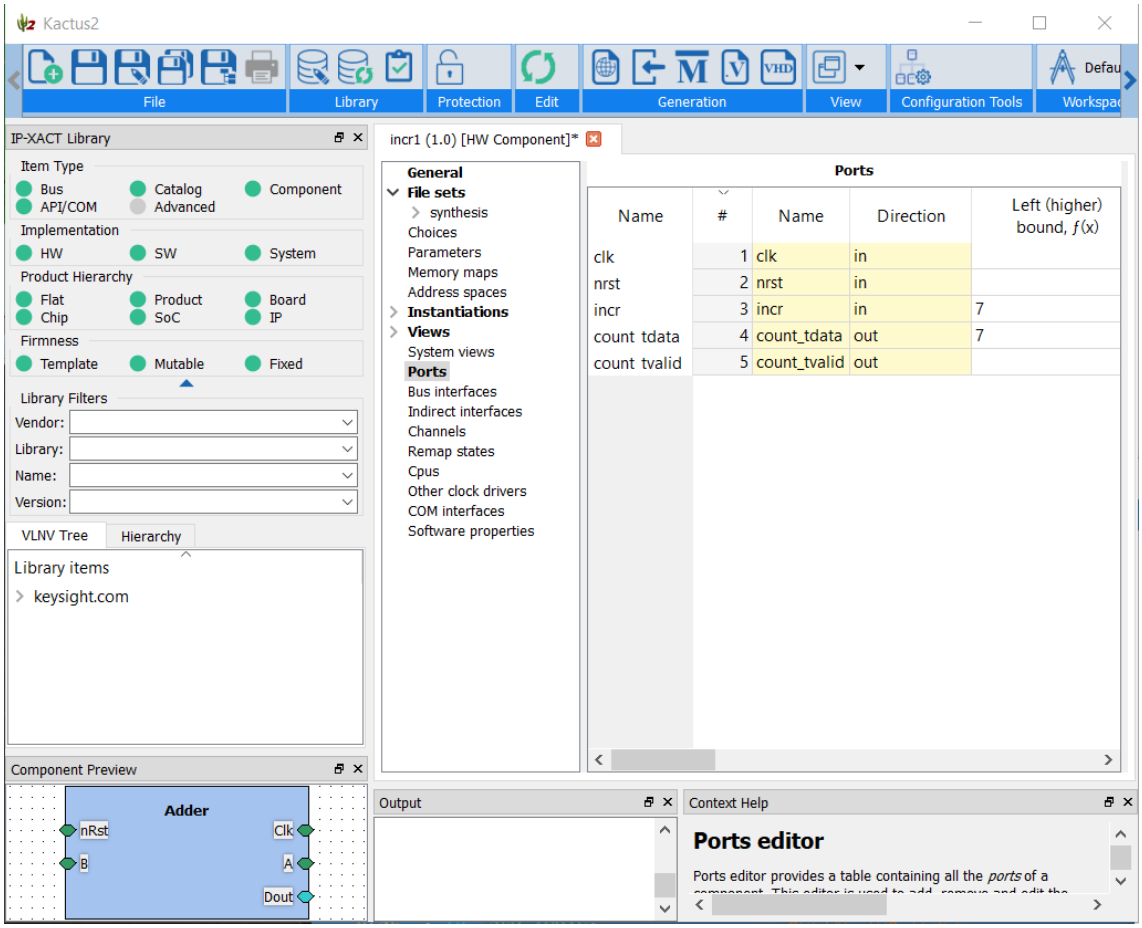
By default, Kactus2 includes all the files in the source directory. In the upper-right pane, click on **File sets/synthesis** to bring up the file set editor:



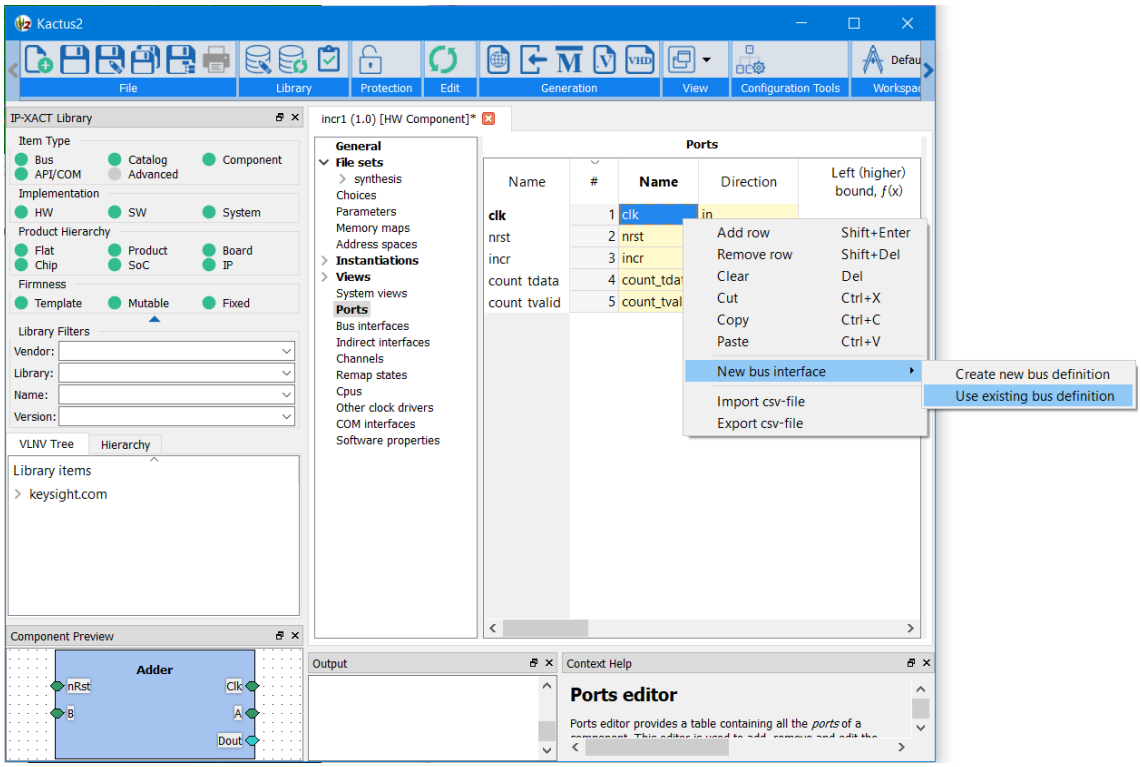
If the list of files contains files that are not part of desired IP blocks source, delete them using **right-click/remove row** or **Shift+del**.



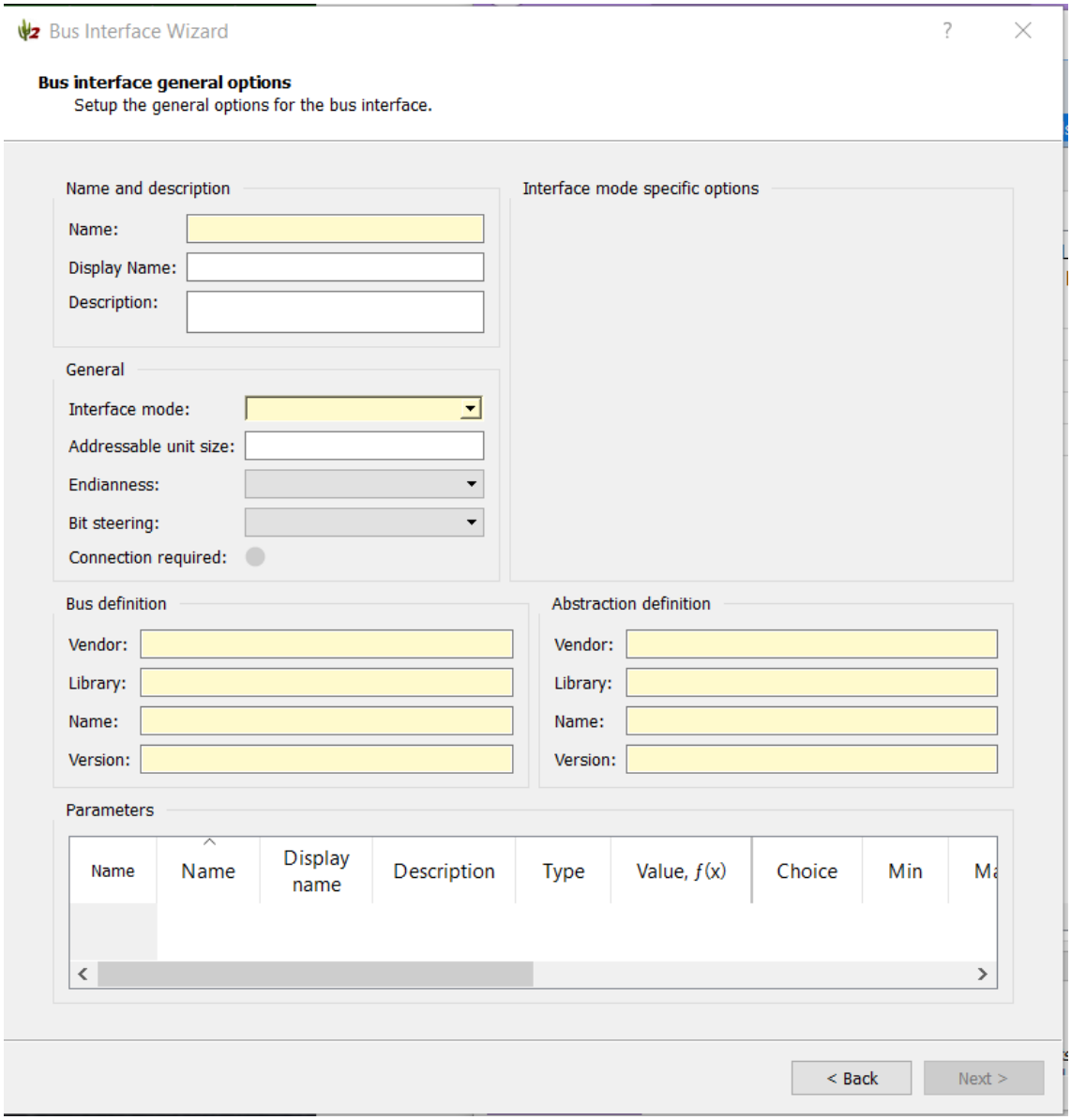
Now that the list of source files has been fixed, it is time to assign ports to logical *Bus Interfaces*. This allows PathWave FPGA to more easily connect interfaces between IP blocks. Click on **Ports** to bring up the Ports Editor. This should show all the ports that were read from the source HDL file and shows things like the direction (in or out), the width, and the index bounds for vectors:



To assign a single port to an interface, select the port in the yellow box under the **Name** column, right-click and select **New bus interface/Use existing bus definition**.



This will bring up the **Bus Interface Wizard**. This wizard will be used to assign all the ports to interfaces:

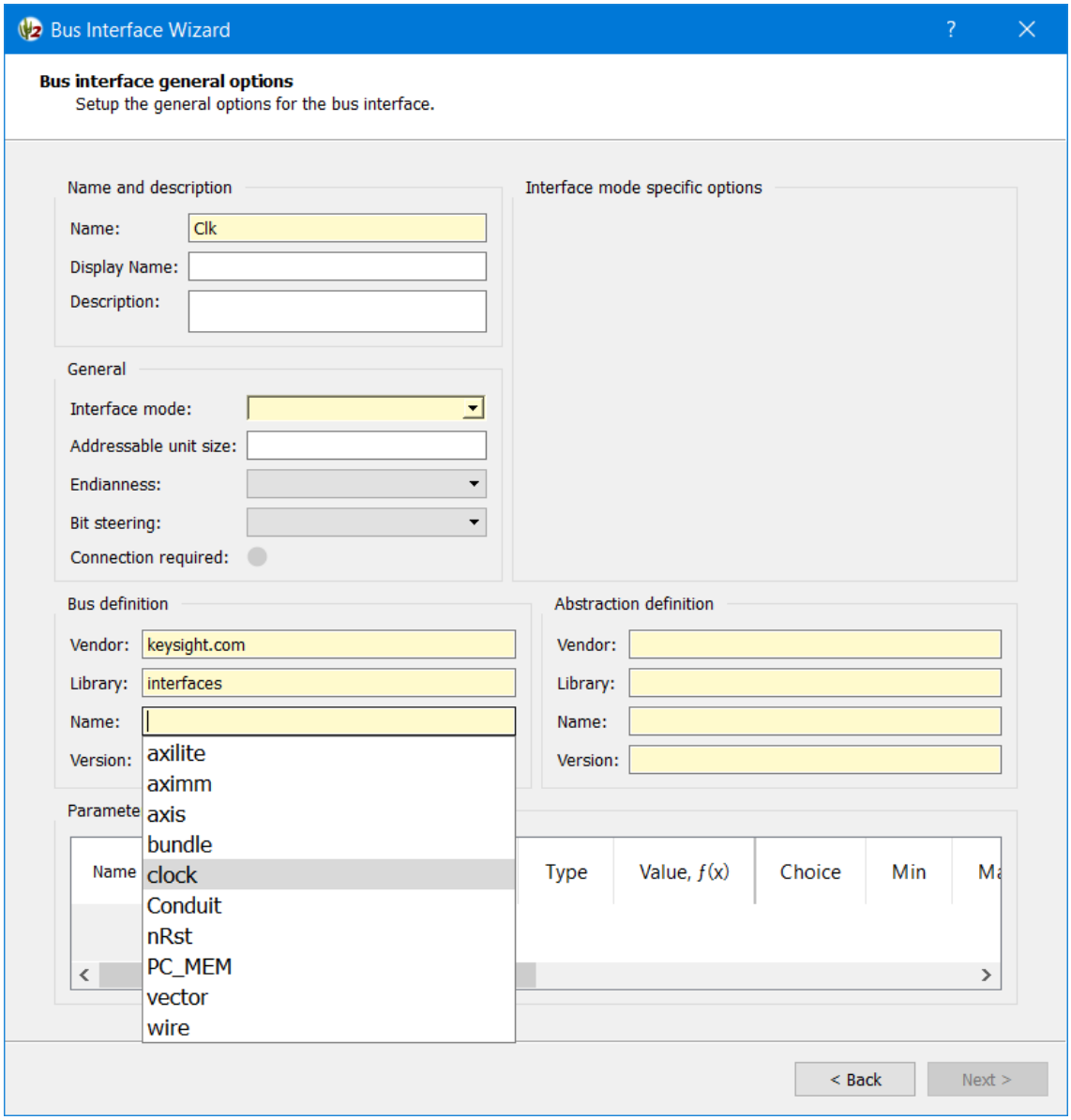


At the **Introduction** screen, click **Next** to bring up the **Bus interface general options** page.

Note that the boxes shaded in yellow are required fields that need to be filled out. White boxes are optional fields. Select the **Name:** box and enter a name for the interface. This is typically the name of the port, though it does not have to be. Next the **Bus definition** and **Abstraction definition** fields need to be filled out. Data entry can be speeded up by using the tab key.

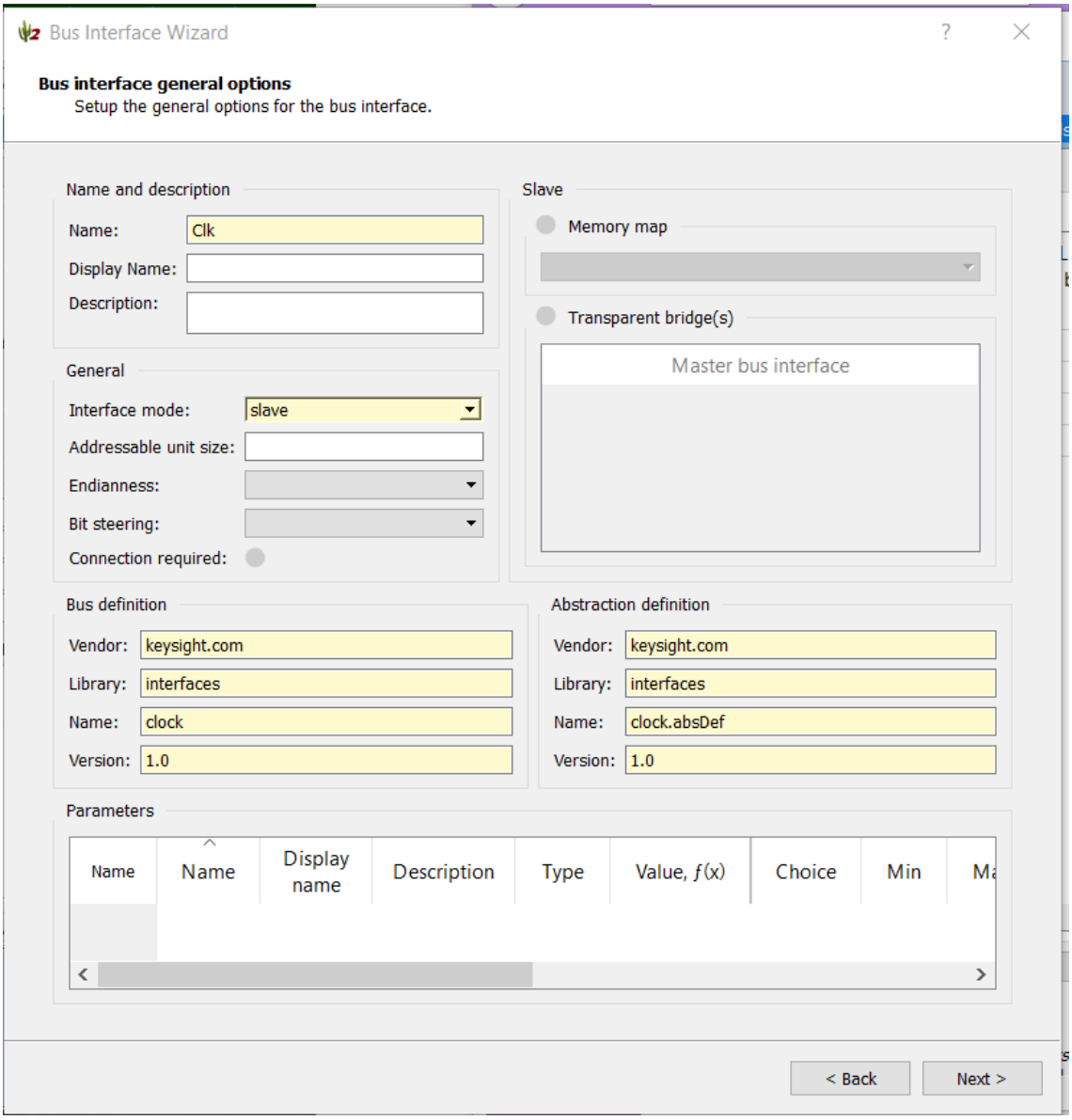
Click on **Vendor:** and *keysight.com* should show up as a suggested entry. Press the tab key to select *keysight.com* and move on to the next field, **Library**. Here, *interfaces* should show up as a suggested entry. Press the tab key to select *interfaces* and move on to the **Name:** field. Alternately, click on the *interfaces* entry to select it and then click on the **Name:** entry to advance to that field.

Under **Name:** select the type of interface that this port should be assigned to. In this case the port is a clock signal, so *clock* should be picked:

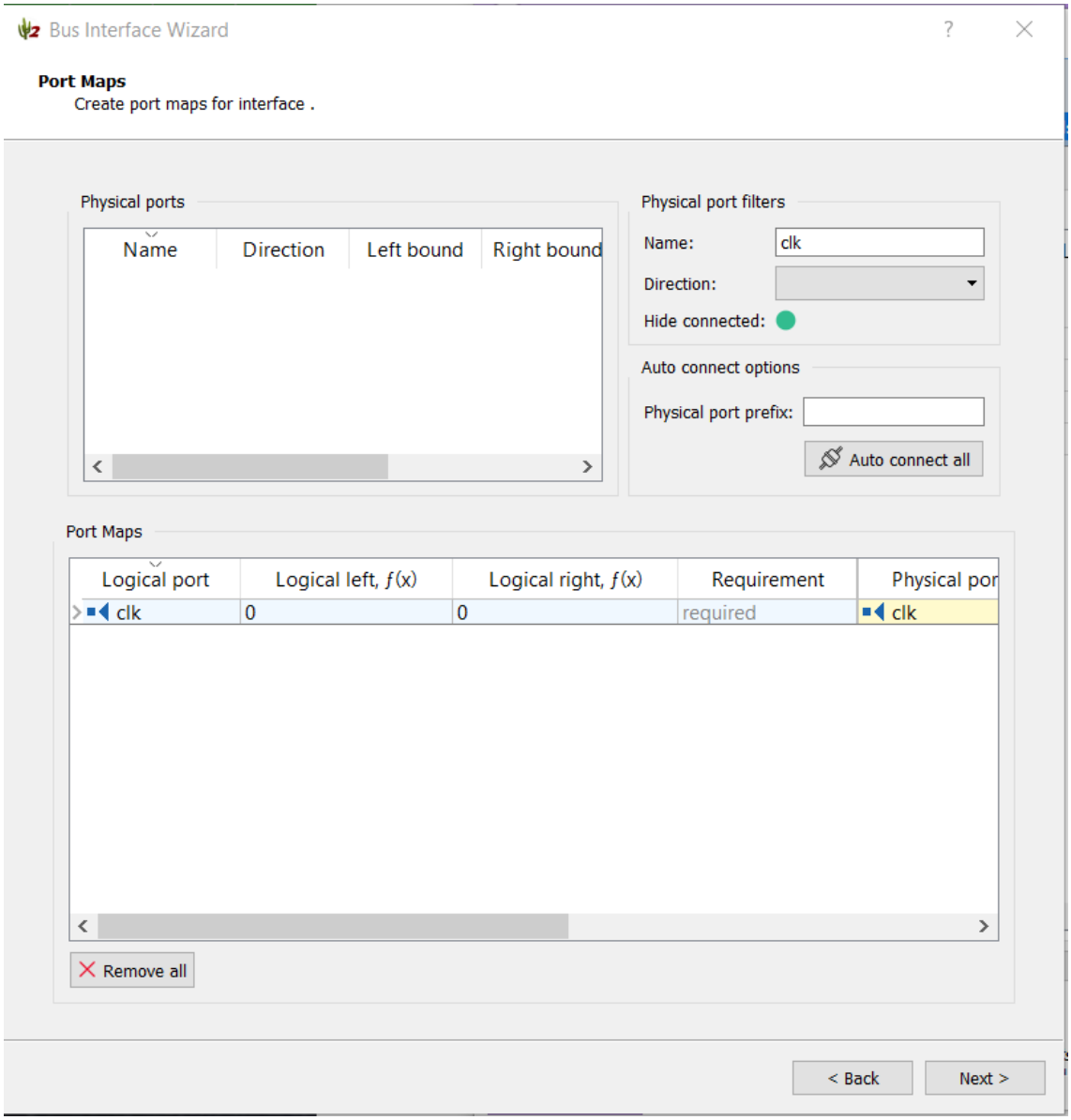


Note that one can speed up the selection by starting to type the name *cl...* to skip down the list more quickly.

After selecting *clock*, press the tab key to select the **Version:**. Press tab four more times to fill out the default **Abstraction definition** fields. The last tab will place the cursor in the **Interface mode:** field. This selects whether the interface is a *master*, meaning it generates the signal, or a *slave*, meaning it consumes the signal. In this case, the clock port is an input port and hence *slave* should be selected:

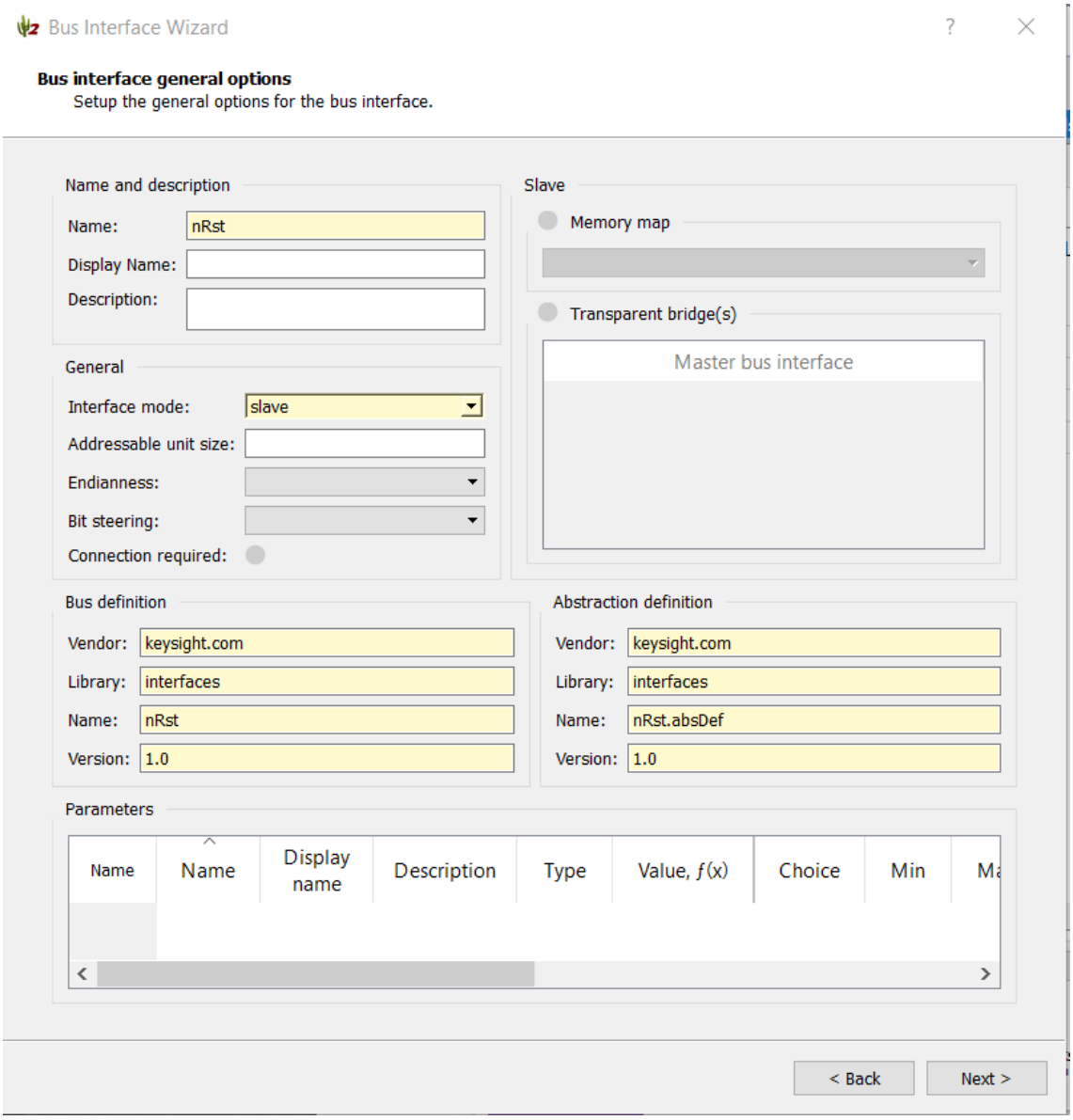


Next the ports need to be assigned to their various roles within the interface. For interfaces with only one port, this is trivial. Click **Next** twice to get to the **Port Maps**. Here the port mapping would be set, but in this case there is only one port so it is automatically filled in:

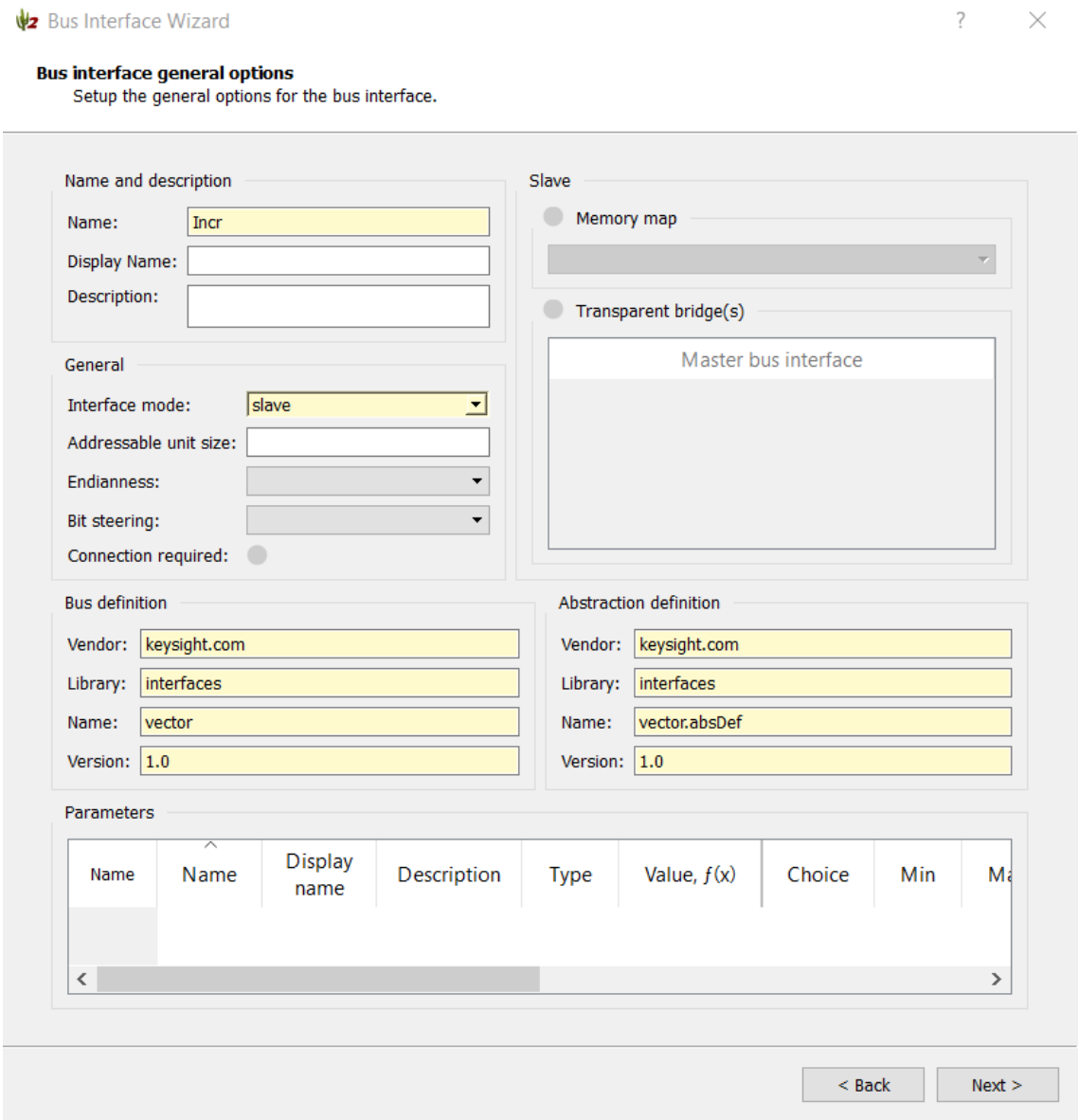


Click **Next** and **Finish** to complete the definition of this interface.

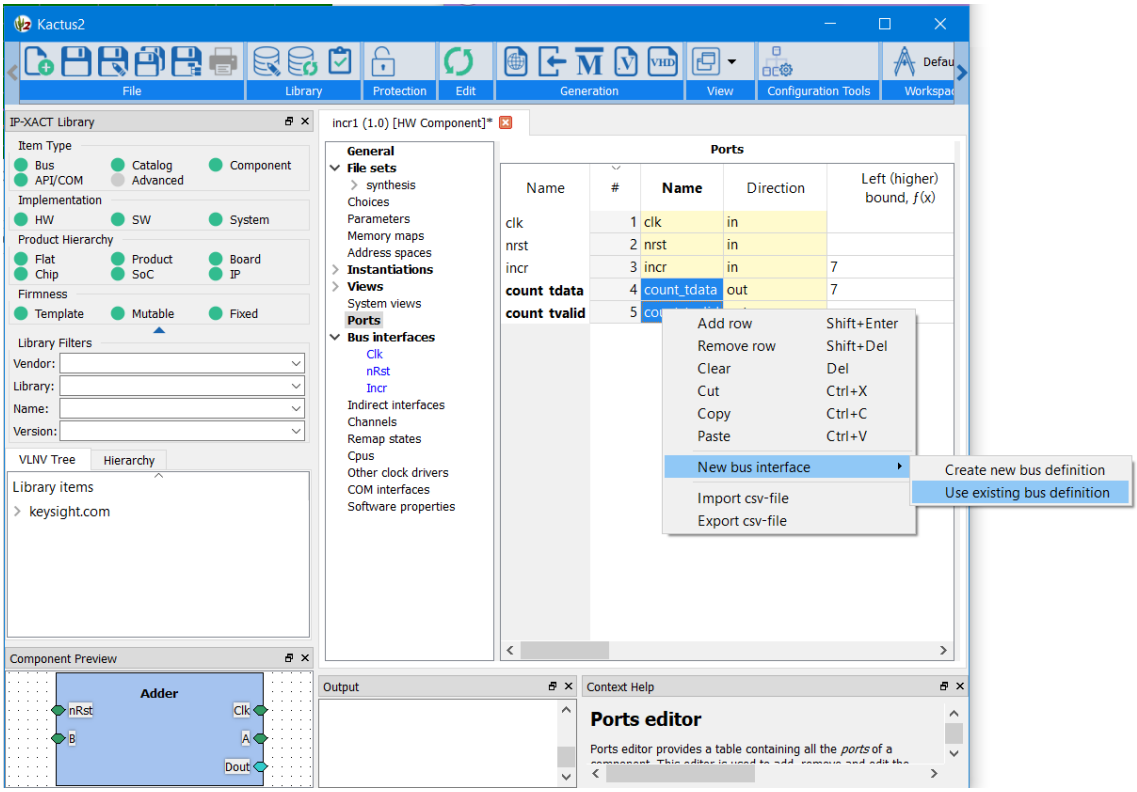
Repeat this process with the *nrst* port using the *nRst* interface:



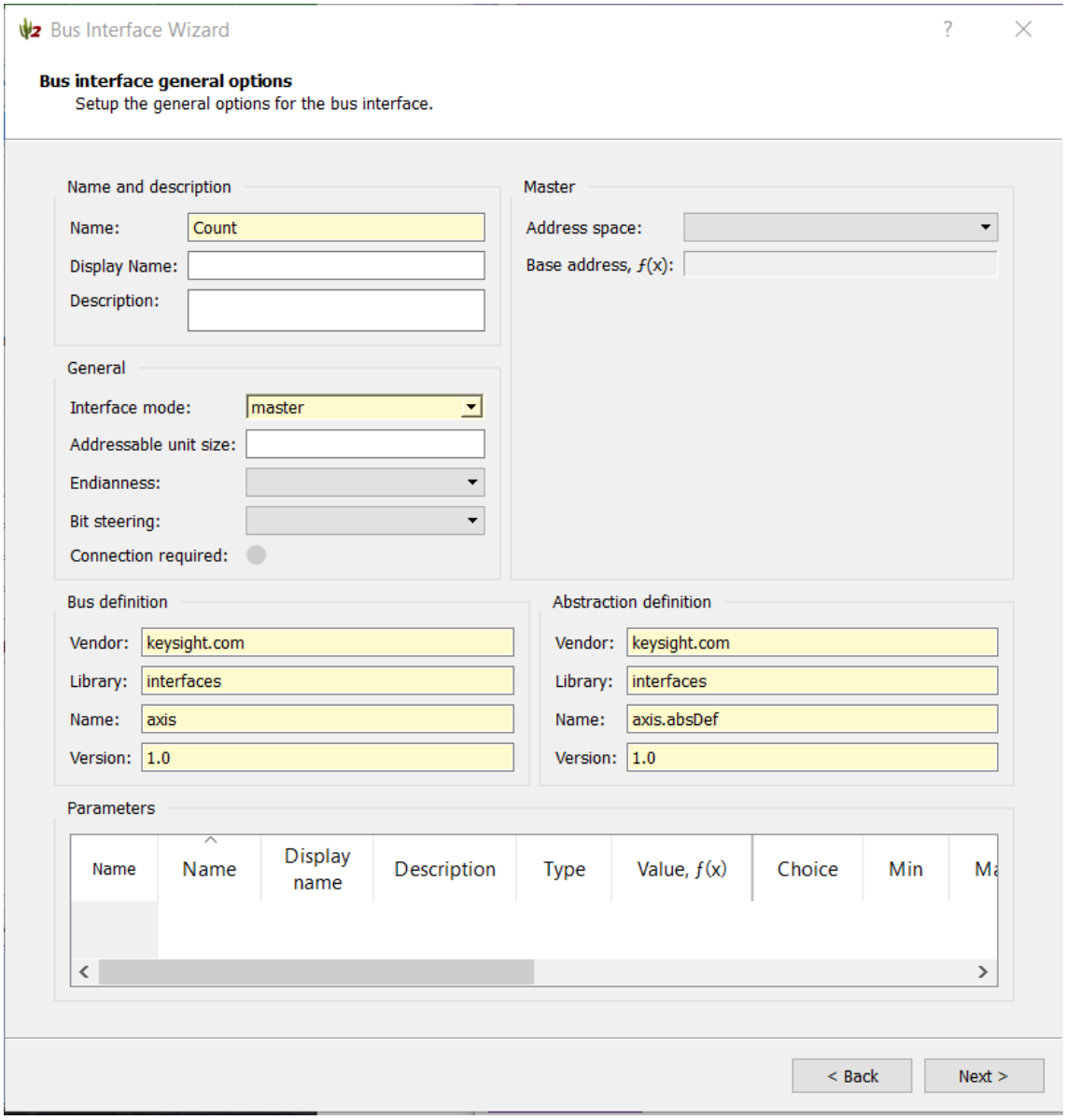
This process works for logic vectors too. Select the *incr* port and configure it as a *vector* interface:



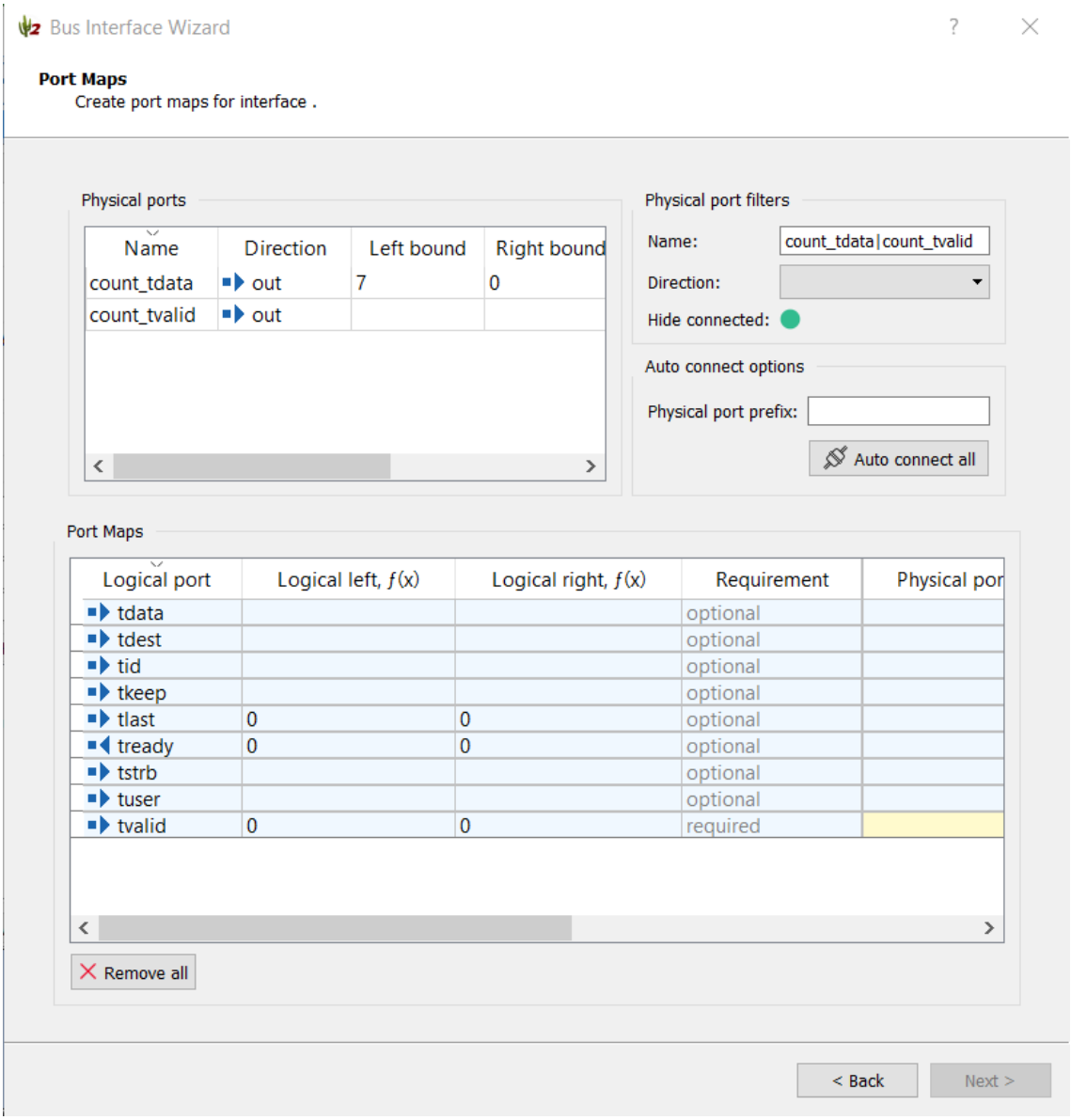
The interface for the *count* ports is a little different. In this case there are more than one port associated with the one interface. This is a case of an AXI-streaming interface consisting of both the *count_tdata* and *count_tvalid* ports. Select both the *count_tdata* and *count_tvalid* ports, right-click, and select the **New bus interface/Use existing bus definition** as before (note: it is okay to select more than the ports associated with the interface as long as all the ports that are associated with the interface are selected):



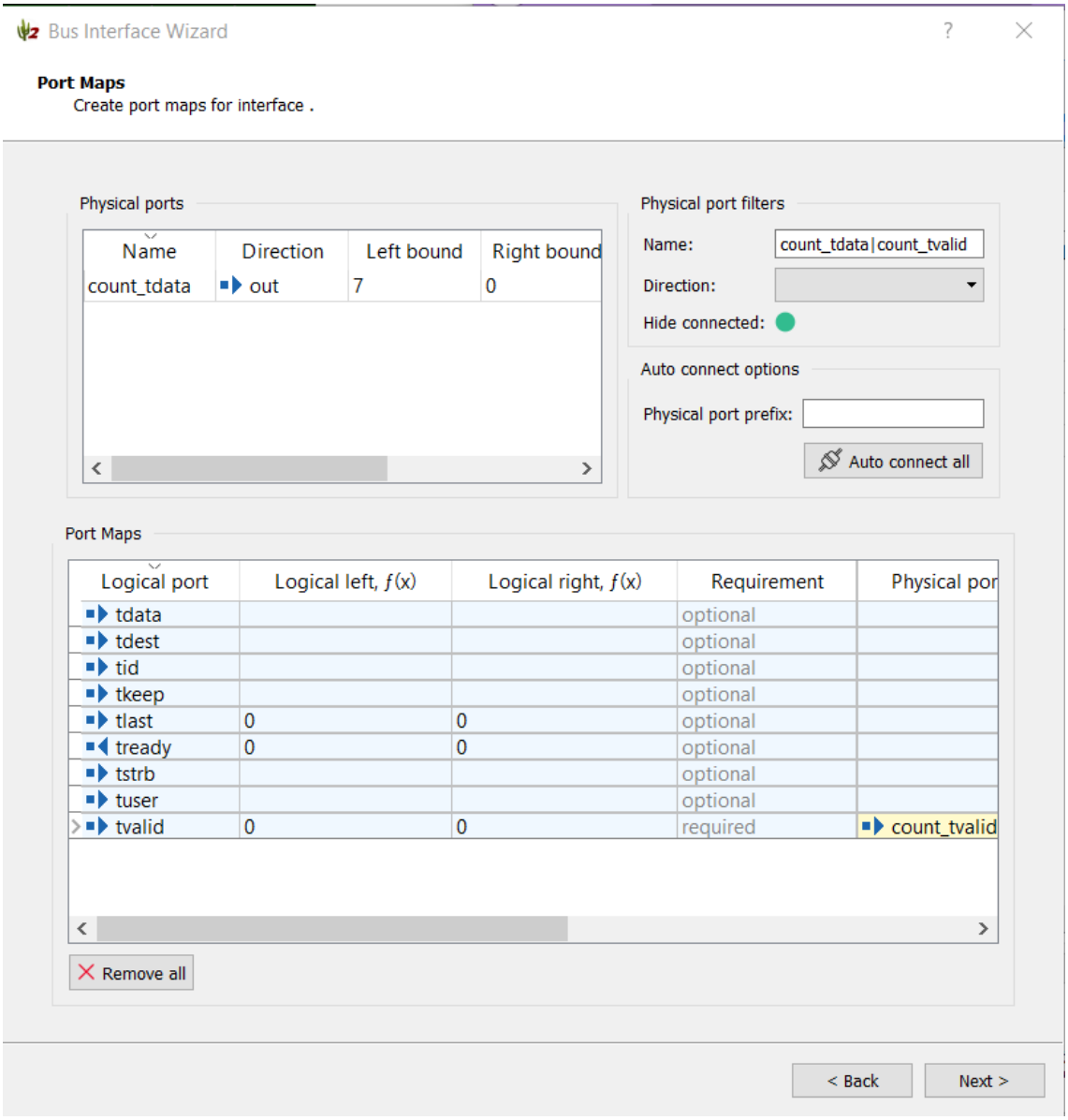
Fill in the **Bus interface general options** page using the *axis* interface name. In this case, the ports are outputs and the interface is a *master*:



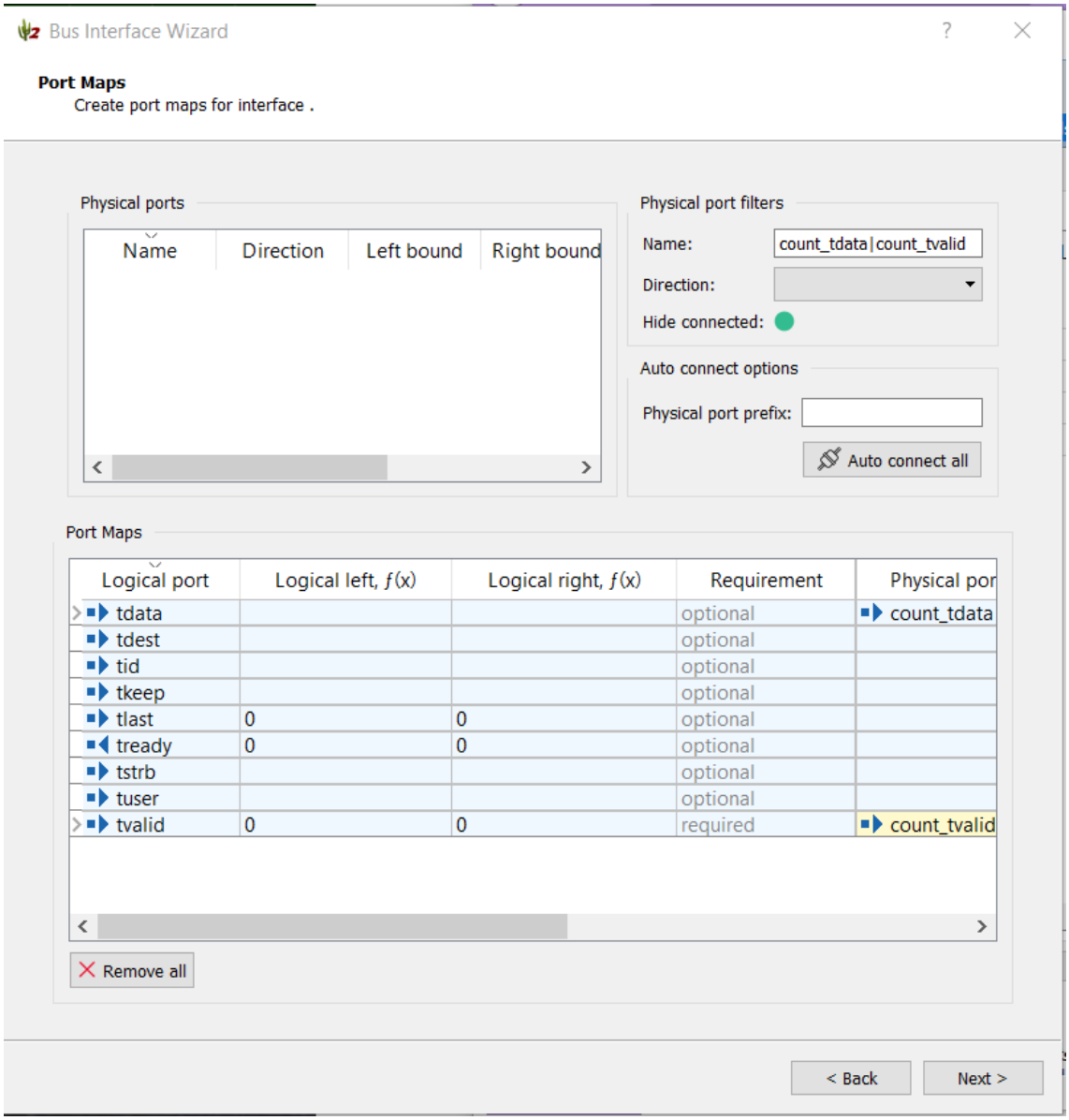
Now at the **Port Maps** page, one sees that there are multiple logical ports listed. Note that only the *tvalid* line has a yellow tinted box. That is the only port required by the AXI streaming spec. All the other ports are optional. Of these, this IP block only uses the *tdata* logical port.



To assign the *count_tvalid* physical port to the *tvalid* logical port, select *count_valid* and drag it down to the *tvalid* row:



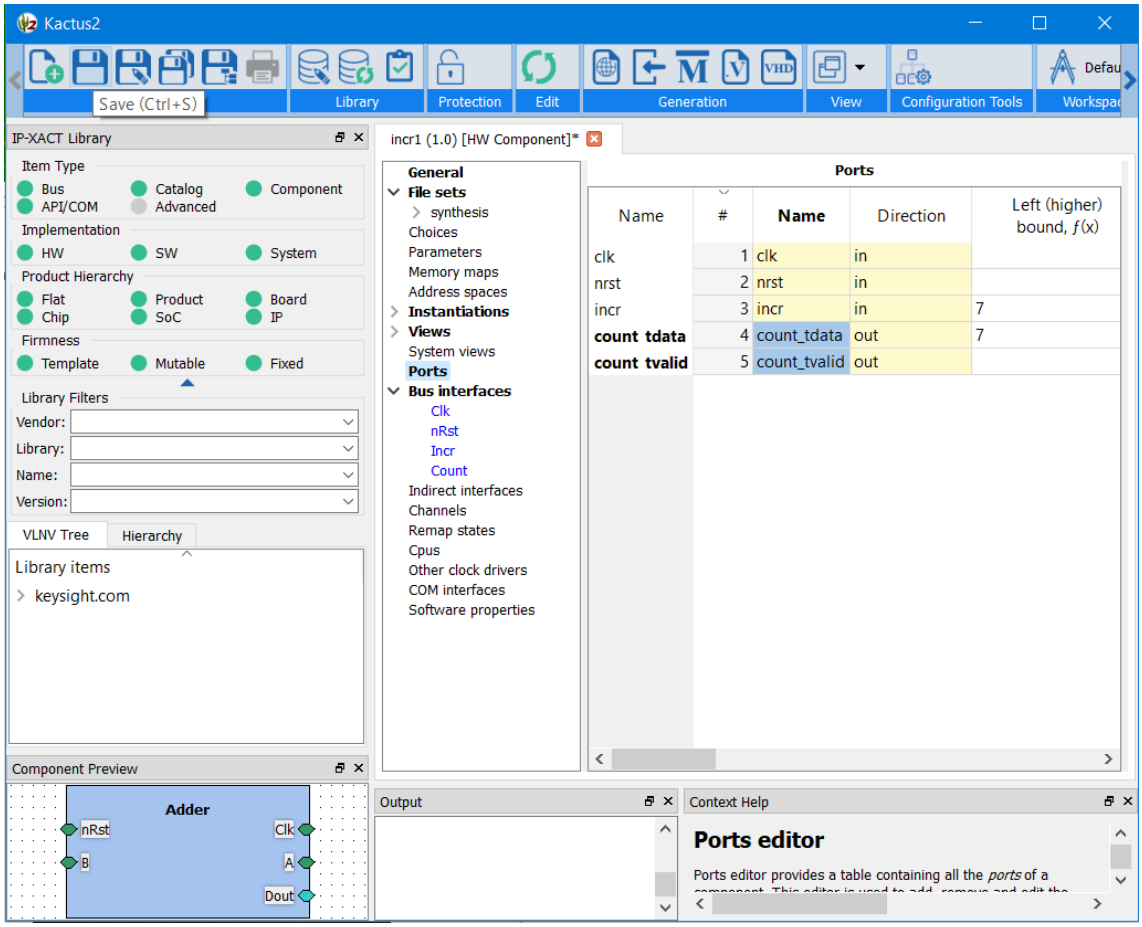
Likewise, drag the *count_tdata* physical port down to the *tdata* line:



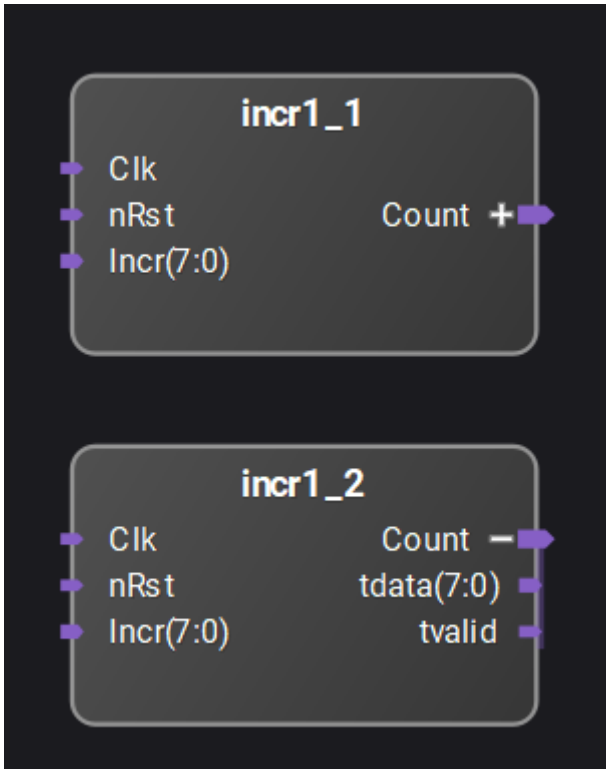
Click **Next** and **Finish** to complete the interface. At this point one can see that under the **Bus interfaces** section, all four of the IP block's interfaces are now listed.

The definition of the IP block's interfaces is complete. If any of the entries in the middle pane are red (none are in this case), that would indicate that there is an error with that entry. Select it and fix any errors until none of the entries are red.

Click the Save icon (or type Ctrl+S) to save the IP-XACT file.



The description of this block's interfaces is now complete and PathWave FPGA can now use this interface information to allow easier connections to other blocks.



In this screen capture from PathWave FPGA, the instance incr1_1 is shown with the *Count* interface collapsed. The internal ports that make up that interface are not shown and the

interface can be connected to other compatible interfaces with one connection. The instance incr1_2 is shown with the *Count* interface expanded to show the internal ports that make up that interface. The entire interface can be connected with one connection by using the *Count* port or the individual ports within the interface can be connected separately if desired.

The generated IP-XACT is

Code Block 7 incr1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ipxact:component xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014"
xmlns:kactus2="http://kactus2.cs.tut.fi"
xsi:schemaLocation="http://www.accellera.org/XMLSchema/IPXACT/1685-2014
http://www.accellera.org/XMLSchema/IPXACT/1685-2014/index.xsd">
 <ipxact:vendor>keysight.com</ipxact:vendor>
 <ipxact:library>flat</ipxact:library>
 <ipxact:name>incr1</ipxact:name>
 <ipxact:version>1.0</ipxact:version>
 <ipxact:busInterfaces>
    <ipxact:busInterface>
       <ipxact:name>Clk</ipxact:name>
       <ipxact:busType vendor="keysight.com" library="interfaces"
name="clock" version="1.0"/>
       <ipxact:abstractionTypes>
          <ipxact:abstractionType>
             <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="clock.absDef" version="1.0"/>
             <ipxact:portMaps>
                <ipxact:portMap>
                   <ipxact:logicalPort>
                      <ipxact:name>clk</ipxact:name>
                      <ipxact:range>
                         <ipxact:left>0</ipxact:left>
                         <ipxact:right>0</ipxact:right>
                      </ipxact:range>
                   </ipxact:logicalPort>
                   <ipxact:physicalPort>
                      <ipxact:name>clk</ipxact:name>
                      <ipxact:partSelect>
                         <ipxact:range>
                            <ipxact:left>0</ipxact:left>
                            <ipxact:right>0</ipxact:right>
                         </ipxact:range>
                      </ipxact:partSelect>
                   </ipxact:physicalPort>
                </ipxact:portMap>
             </ipxact:portMaps>
          </ipxact:abstractionType>
       </ipxact:abstractionTypes>
       <ipxact:slave/>
    </ipxact:busInterface>
    <ipxact:busInterface>
       <ipxact:name>nRst</ipxact:name>
       <ipxact:busType vendor="keysight.com" library="interfaces"
name="nRst" version="1.0"/>
       <ipxact:abstractionTypes>
          <ipxact:abstractionType>
             <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="nRst.absDef" version="1.0"/>
             <ipxact:portMaps>
                <ipxact:portMap>
                   <ipxact:logicalPort>
                      <ipxact:name>nRst</ipxact:name>
                      <ipxact:range>
                         <ipxact:left>0</ipxact:left>
```

```
                      <ipxact:right>0</ipxact:right>
                   </ipxact:range>
                </ipxact:logicalPort>
                <ipxact:physicalPort>
                   <ipxact:name>nrst</ipxact:name>
                   <ipxact:partSelect>
                      <ipxact:range>
                         <ipxact:left>0</ipxact:left>
                         <ipxact:right>0</ipxact:right>
                      </ipxact:range>
                   </ipxact:partSelect>
                </ipxact:physicalPort>
             </ipxact:portMap>
          </ipxact:portMaps>
       </ipxact:abstractionType>
    </ipxact:abstractionTypes>
    <ipxact:slave/>
 </ipxact:busInterface>
 <ipxact:busInterface>
    <ipxact:name>Incr</ipxact:name>
    <ipxact:busType vendor="keysight.com" library="interfaces"
name="vector" version="1.0"/>
    <ipxact:abstractionTypes>
       <ipxact:abstractionType>
          <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="vector.absDef" version="1.0"/>
          <ipxact:portMaps>
             <ipxact:portMap>
                <ipxact:logicalPort>
                   <ipxact:name>Signal</ipxact:name>
                   <ipxact:range>
                      <ipxact:left>7</ipxact:left>
                      <ipxact:right>0</ipxact:right>
                   </ipxact:range>
                </ipxact:logicalPort>
                <ipxact:physicalPort>
                   <ipxact:name>incr</ipxact:name>
                   <ipxact:partSelect>
                      <ipxact:range>
                         <ipxact:left>7</ipxact:left>
                         <ipxact:right>0</ipxact:right>
                      </ipxact:range>
                   </ipxact:partSelect>
                </ipxact:physicalPort>
             </ipxact:portMap>
          </ipxact:portMaps>
       </ipxact:abstractionType>
    </ipxact:abstractionTypes>
    <ipxact:slave/>
 </ipxact:busInterface>
 <ipxact:busInterface>
    <ipxact:name>Count</ipxact:name>
    <ipxact:busType vendor="keysight.com" library="interfaces"
name="axis" version="1.0"/>
    <ipxact:abstractionTypes>
       <ipxact:abstractionType>
          <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="axis.absDef" version="1.0"/>
          <ipxact:portMaps>
             <ipxact:portMap>
                <ipxact:logicalPort>
                   <ipxact:name>tdata</ipxact:name>
                </ipxact:logicalPort>
                <ipxact:physicalPort>
                   <ipxact:name>count_tdata</ipxact:name>
                </ipxact:physicalPort>
```

```
              </ipxact:portMap>
              <ipxact:portMap>
                 <ipxact:logicalPort>
                    <ipxact:name>tvalid</ipxact:name>
                 </ipxact:logicalPort>
                 <ipxact:physicalPort>
                    <ipxact:name>count_tvalid</ipxact:name>
                 </ipxact:physicalPort>
              </ipxact:portMap>
           </ipxact:portMaps>
        </ipxact:abstractionType>
     </ipxact:abstractionTypes>
     <ipxact:master/>
   </ipxact:busInterface>
 </ipxact:busInterfaces>
 <ipxact:model>
   <ipxact:views>
     <ipxact:view>
        <ipxact:name>flat_vhdl</ipxact:name>
        <ipxact:envIdentifier>VHDL:Kactus2:</ipxact:envIdentifier>

 <ipxact:componentInstantiationRef>vhdl_implementation</ipxact:componentI
nstantiationRef>
     </ipxact:view>
   </ipxact:views>
   <ipxact:instantiations>
     <ipxact:componentInstantiation>
        <ipxact:name>vhdl_implementation</ipxact:name>
        <ipxact:language>VHDL</ipxact:language>
        <ipxact:moduleName>incr1</ipxact:moduleName>
        <ipxact:architectureName>Behavioral</ipxact:architectureName>
        <ipxact:fileSetRef>
           <ipxact:localName>src</ipxact:localName>
        </ipxact:fileSetRef>
     </ipxact:componentInstantiation>
   </ipxact:instantiations>
   <ipxact:ports>
     <ipxact:port>
        <ipxact:name>clk</ipxact:name>
        <ipxact:wire>
           <ipxact:direction>in</ipxact:direction>
           <ipxact:wireTypeDefs>
              <ipxact:wireTypeDef>
                 <ipxact:typeName>std_logic</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
              </ipxact:wireTypeDef>
           </ipxact:wireTypeDefs>
        </ipxact:wire>
     </ipxact:port>
     <ipxact:port>
        <ipxact:name>nrst</ipxact:name>
        <ipxact:wire>
           <ipxact:direction>in</ipxact:direction>
           <ipxact:wireTypeDefs>
              <ipxact:wireTypeDef>
                 <ipxact:typeName>std_logic</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
              </ipxact:wireTypeDef>
           </ipxact:wireTypeDefs>
        </ipxact:wire>
     </ipxact:port>
     <ipxact:port>
        <ipxact:name>incr</ipxact:name>
        <ipxact:wire>
```

```
            <ipxact:direction>in</ipxact:direction>
            <ipxact:vectors>
               <ipxact:vector>
                  <ipxact:left>7</ipxact:left>
                  <ipxact:right>0</ipxact:right>
               </ipxact:vector>
            </ipxact:vectors>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>std_logic_vector</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>count_tdata</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>out</ipxact:direction>
            <ipxact:vectors>
               <ipxact:vector>
                  <ipxact:left>7</ipxact:left>
                  <ipxact:right>0</ipxact:right>
               </ipxact:vector>
            </ipxact:vectors>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>std_logic_vector</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>count_tvalid</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>out</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>std_logic</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
   </ipxact:ports>
 </ipxact:model>
 <ipxact:fileSets>
   <ipxact:fileSet>
      <ipxact:name>src</ipxact:name>
      <ipxact:file>
         <ipxact:name>src/incr1.vhd</ipxact:name>
         <ipxact:fileType>vhdlSource</ipxact:fileType>
         <ipxact:vendorExtensions>

 <kactus2:hash>ef09beac89449e3689558b669252ef520e1a34d8</kactus2:hash>
         </ipxact:vendorExtensions>
      </ipxact:file>
   </ipxact:fileSet>
 </ipxact:fileSets>
 <ipxact:description>Count increments in multiples of
incr</ipxact:description>
 <ipxact:vendorExtensions>
```

```
    <kactus2:author>Keysight</kactus2:author>
    <kactus2:version>3,5,0,0</kactus2:version>
    <kactus2:kts_attributes>
       <kactus2:kts_productHier>Flat</kactus2:kts_productHier>
       <kactus2:kts_implementation>HW</kactus2:kts_implementation>
       <kactus2:kts_firmness>Mutable</kactus2:kts_firmness>
    </kactus2:kts_attributes>
 </ipxact:vendorExtensions>
</ipxact:component>
```

Import HDL with parameterized bus widths using IP-XACT

IP-XACT or IEEE 1685-2014 is an XML specification for describing (among other things) the interfaces used by an IP block in an FPGA. This tutorial describes the creation of an IP-XACT file for a parameterized IP block written in VHDL. *Parameters* (called *generics* in VHDL) are values that are specified when the IP block is instantiated and can be used to customize the block. This allows one IP block to fill more needs than a non-parameterized block would. For example, instead of requiring multiple IP blocks to support adders of different sizes, one adder block can be parameterized so that the size of the adder can be specified when the block is used.

This tutorial uses a block similar to that which was used in the *IP-XACT Creation Tuturial* with the difference being that this block uses two parameters, *width* which specifies the bit width of the block, and *dir* which specifies whether the block increments or decrements. The process for creating the IP-XACT file is very similar to the case for non-parameterized IP blocks with a few steps added towards the end.

While the IP-XACT file is text and can be manually created in any text editor, it is simpler and easier to use an IP-XACT editor such as Kactus2 (available at http://funbase.cs.tut.fi/). For IP that is parameterized, PathWave FPGA recommends using version 3.5.77 or later.

The HDL IP block has physical ports which are the input and output signals for the IP block. One or more ports can be combined into logical interfaces which describe how the signals interact and connect with other signals. An interface may consist of a single port or even a signal wire. An example of this is a clock interface. Other interfaces, such as the AXI-MM interface, may have dozens of potential ports. By describing which ports constitute a particular interface and which role each port has, the IP-XACT description eases connecting interfaces together. An AXI Master can connect to an AXI Slave with only one connection even though a considerable number of individual ports will be connected in the hardware.

This tutorial will create the IP-XACT for the following parameterized block:

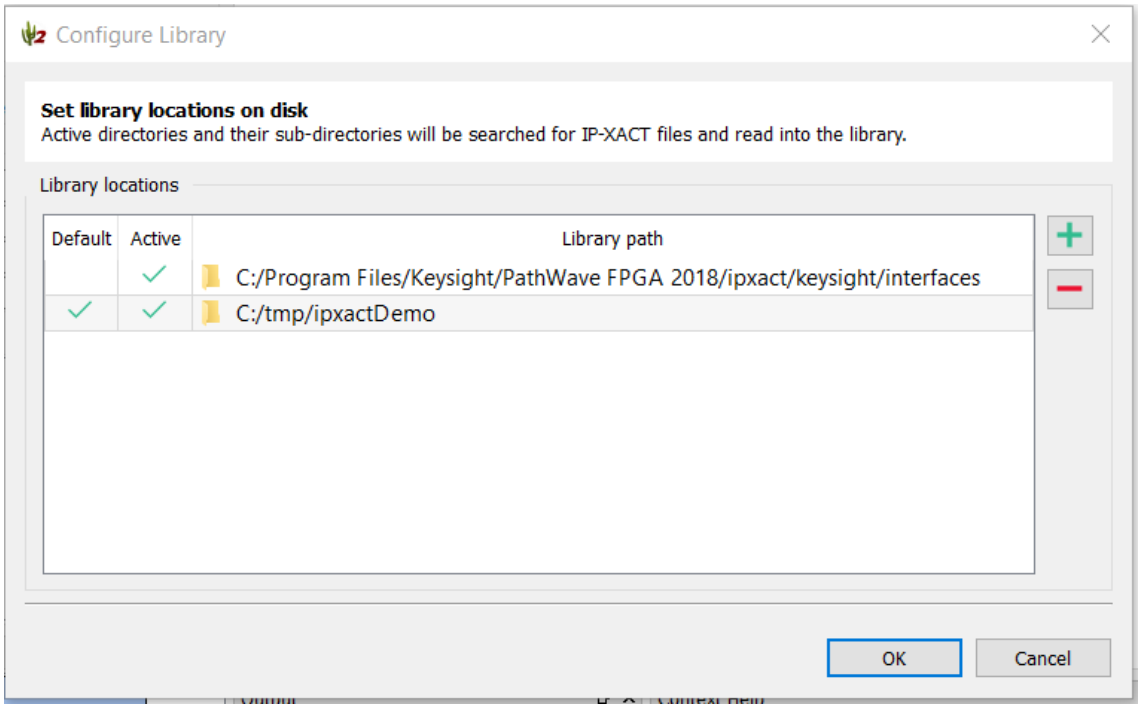
Code Block 8 incr2.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;


entity incr2 is
  generic (
    width : integer := 8;
    dir   : integer := 0);              -- Direction : 0 = up, 1 = down
  port (
```

```
    clk    : in  std_logic;
    nrst   : in  std_logic;              -- Active low reset
    incr   : in  std_logic_vector(width-1 downto 0);
    count_tdata : out std_logic_vector(width-1 downto 0);
    count_tvalid : out std_logic
    );

end incr2;

architecture Behavioral of incr2 is
  signal count : std_logic_vector(width-1 downto 0);
begin  -- Behavioral
  count_tdata <= count;
  count_tvalid <= '1' when (incr /= 0) else '0';
  process(clk)
    begin
      if (nrst = '0') then
        count <= (others => '0');
      else
        if (incr = 1) then
          count <= count + incr;
        else
          count <= count - incr;
        end if;
      end if;
    end process;

end Behavioral;
```

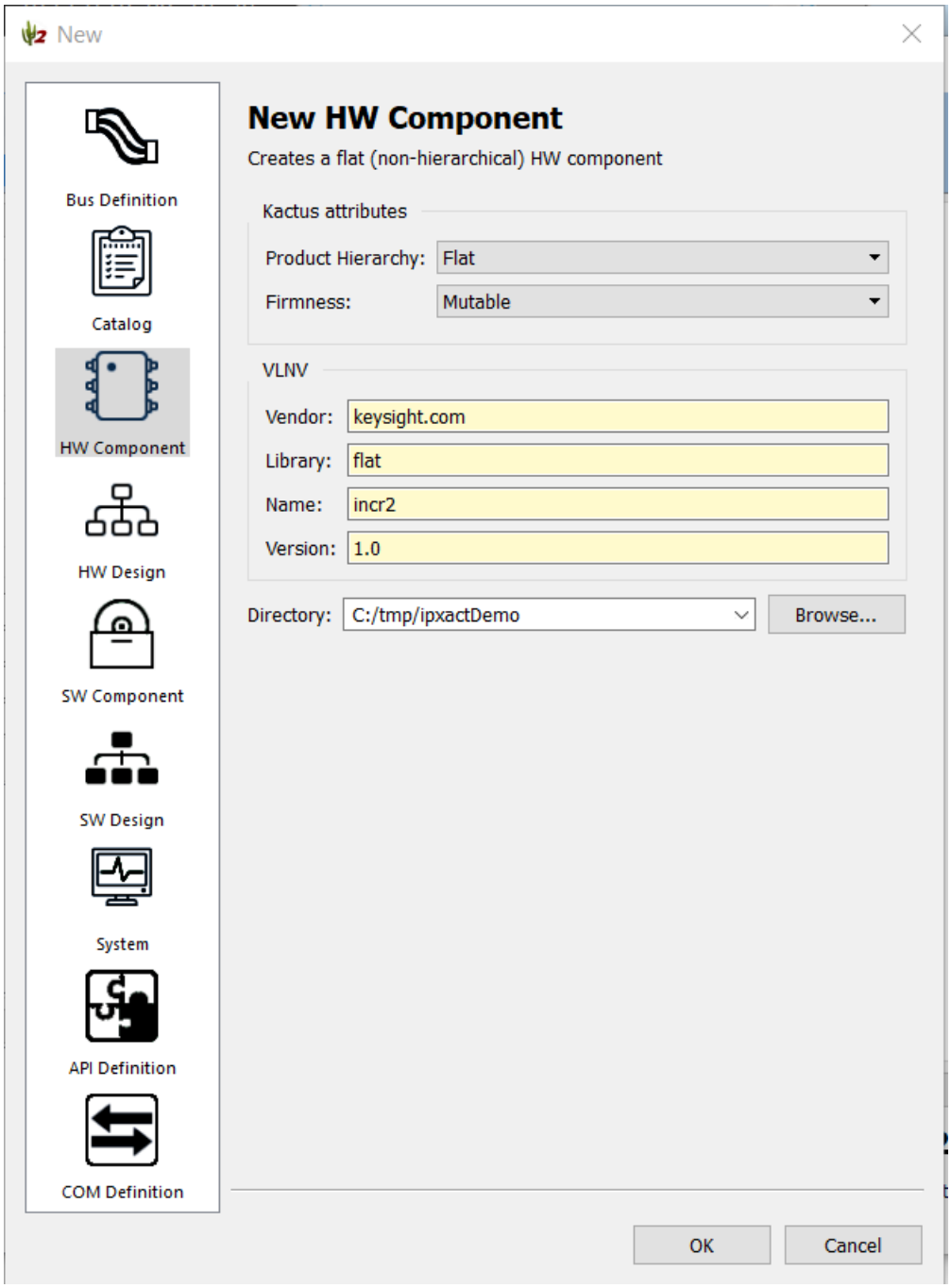
This block will increment or decrement an internal counter based on *dir* and its *incr* input and output the counter value on an AXI-streaming *count* interface. It also has a clock input and an active low nrst input. This HDL file is stored under c:/tmp/ipxactDemo/src/incr2.vhd.

To create IP-XACT for this module, first start Kactus2. Click the **Configure Library** button to set up the libraries. Make sure that the PathWave FPGA interfaces folder as well as the folder for your IP block are both in the library path:

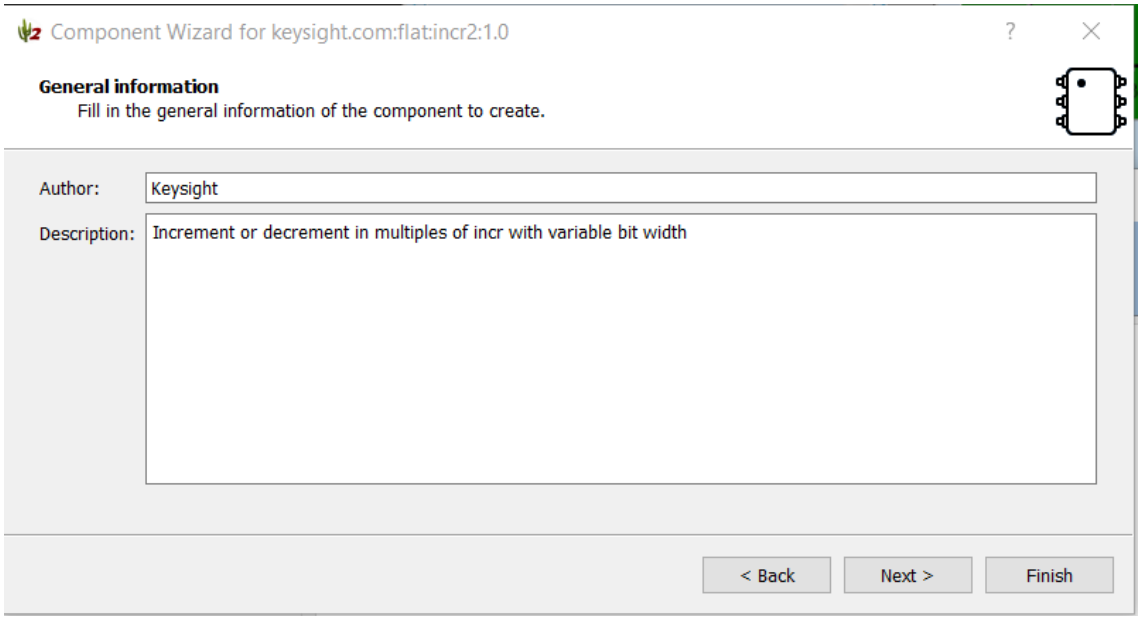


To create the IP-XACT, click the **New** button and select **HW Component.**

In Kactus2, required fields are shown in light yellow while optional fields are shown in white. Enter the **Vendor**, **Library**, **Name**, and **Version** for your IP block. Then click **Browse...** and navigate to the directory with the IP block. In this case, it will be c:/tmp/ipxactDemo. This is where the resulting IP-XACT file will be created:



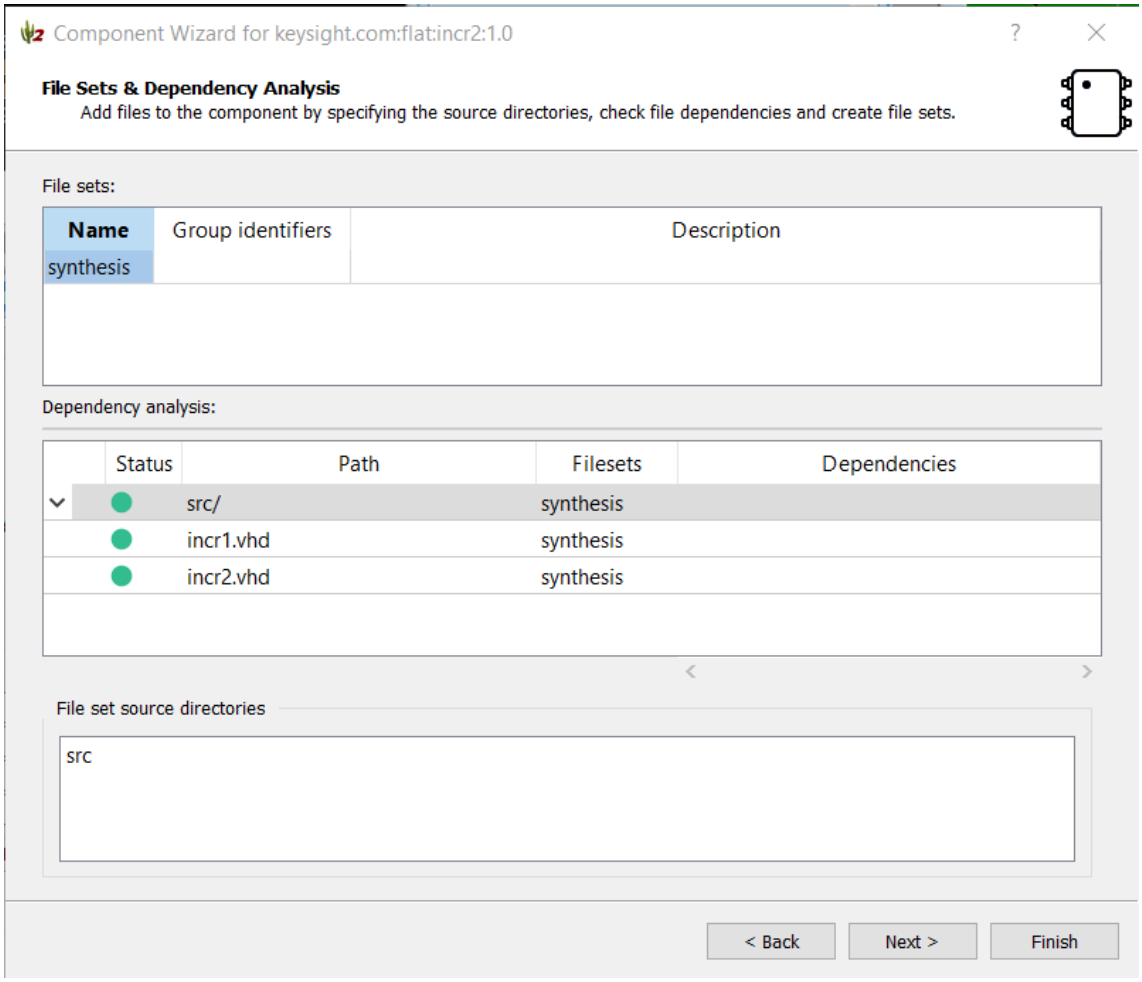
Click **OK** and the Component Wizard is started. Click **Next** to get to the General Information screen, and enter the Author and Description (which are optional):



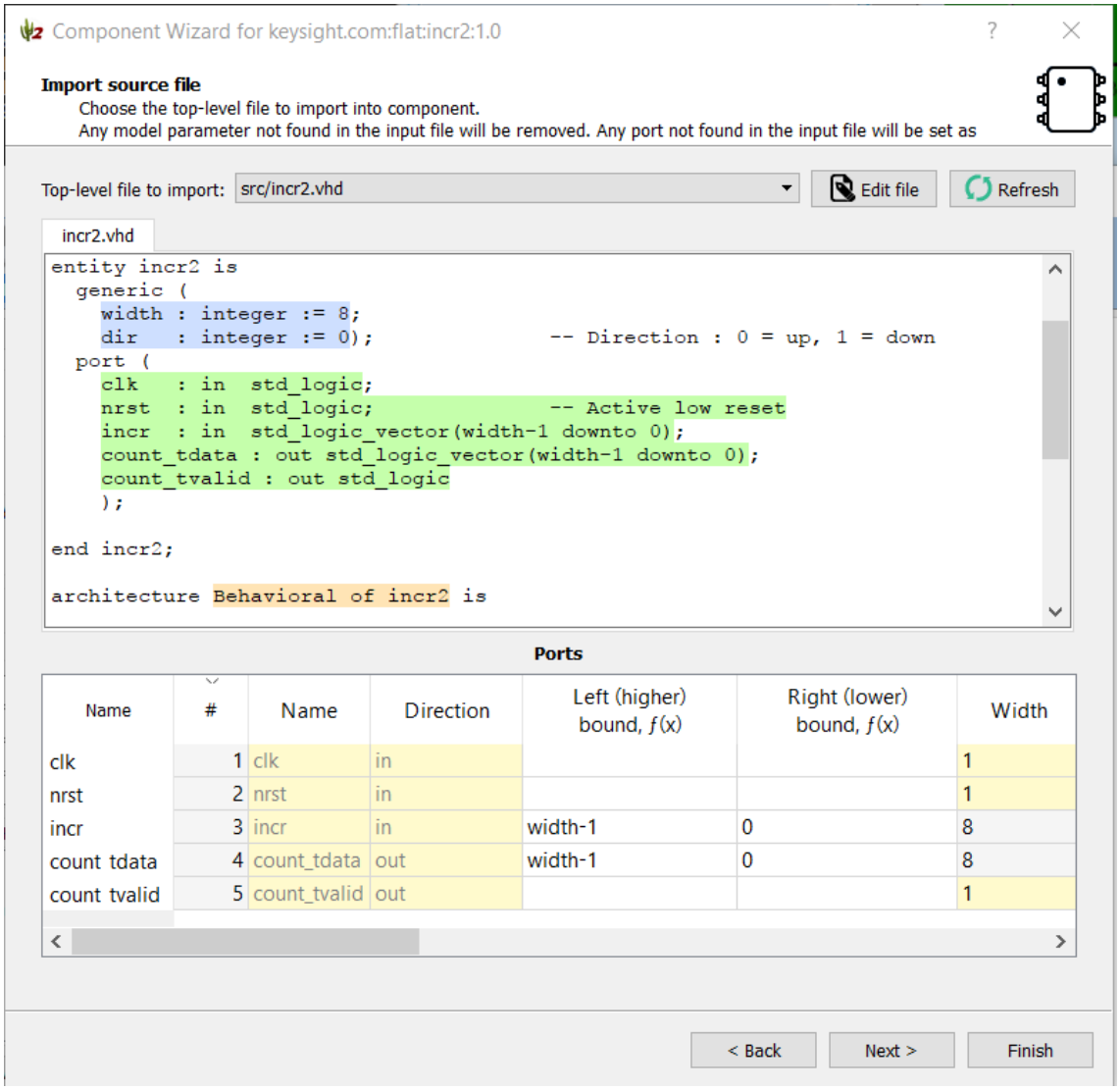
At this point, one could click **Finish** and proceed to enter the IP port information manually, but it is much more convenient to have Kactus2 read the source file and fill in the information automatically.
To do this, the source file folder needs to be set. Click **Next** to get to the File Sets & Dependency Analysis screen and double click in the **File set source directories** box to bring up the selection box.

Select the **src** directory and click **Select Folder.**

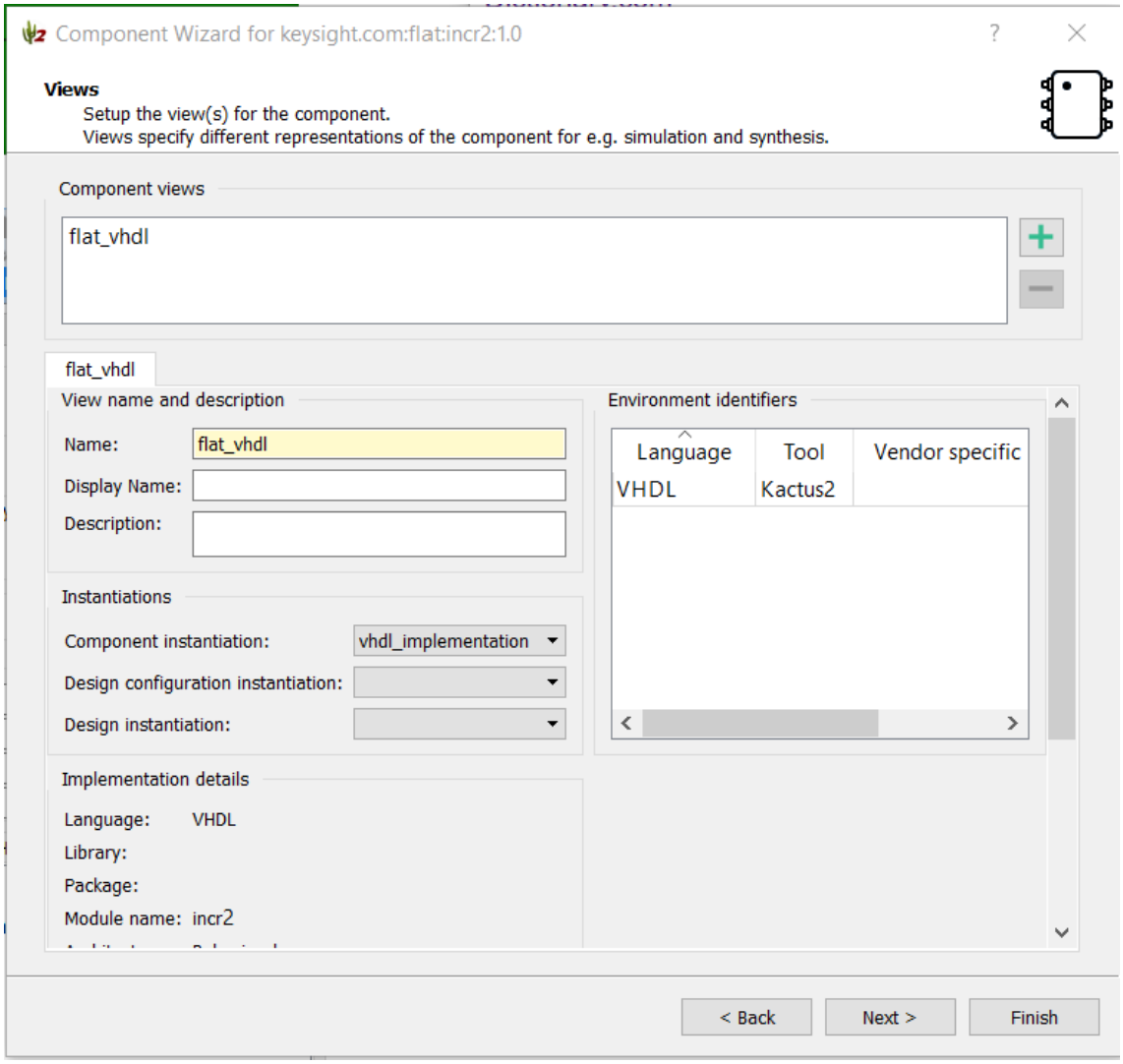
PathWave FPGA uses the **synthesis** fileset for containing the files needed for synthesis. Double click on the **File sets** / **Name** entry and change it to "synthesis":



Click **Next** to advance to the **Import source file** page. This is where the top level source file is specified. Using the pulldown menu for **Top-level file to import:**, select the incr2.vhd file:

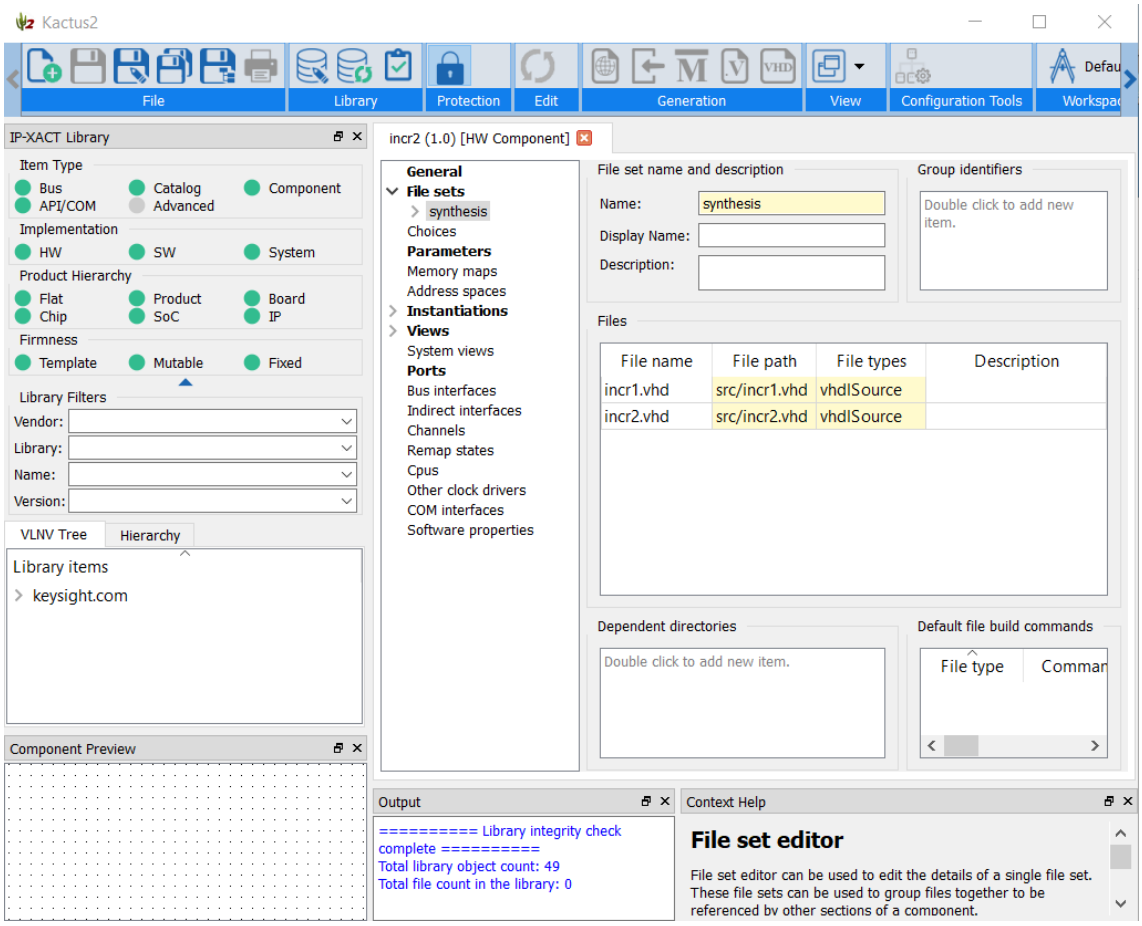


Note that the source file is shown in the middle pane with the detected ports in the lower pane. Click **Next** to advance to the **Views** page:

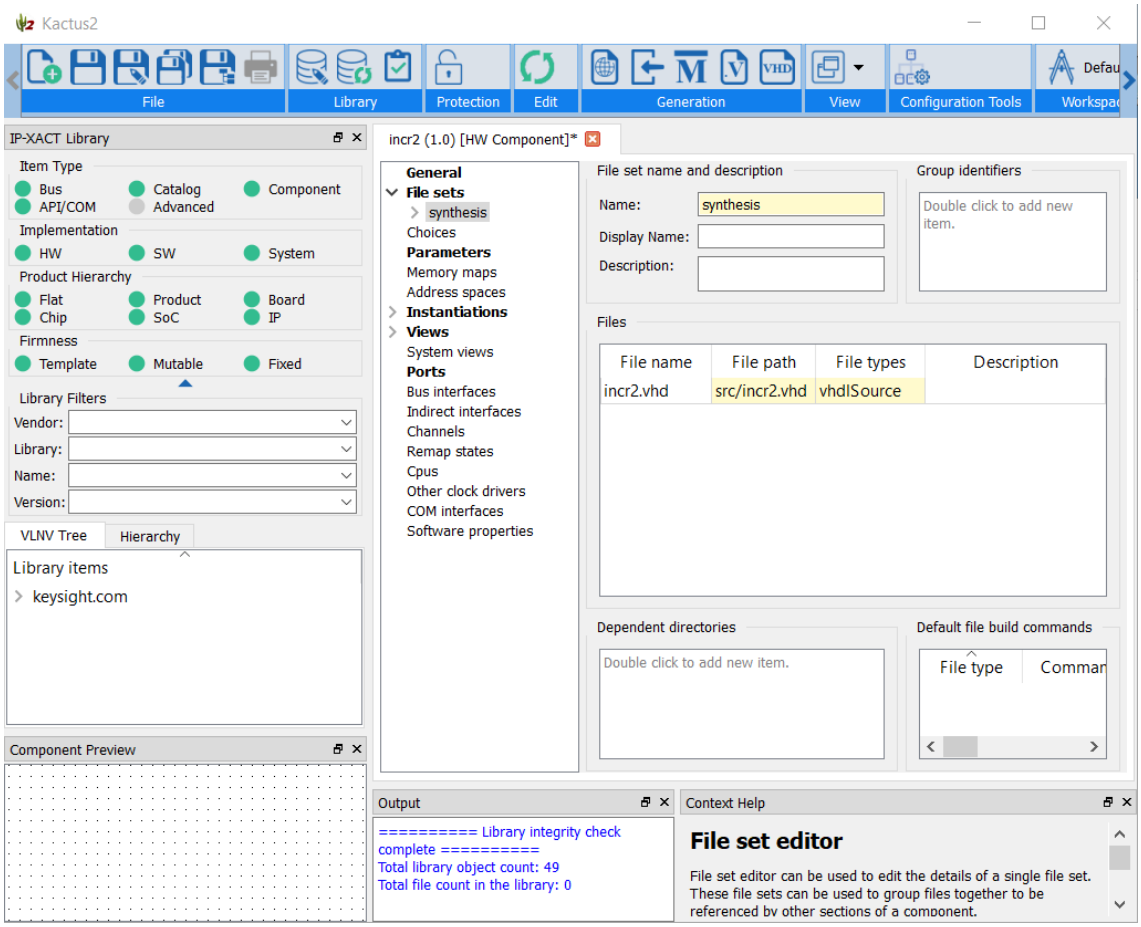


Click **Finish** to complete the Component Wizard.

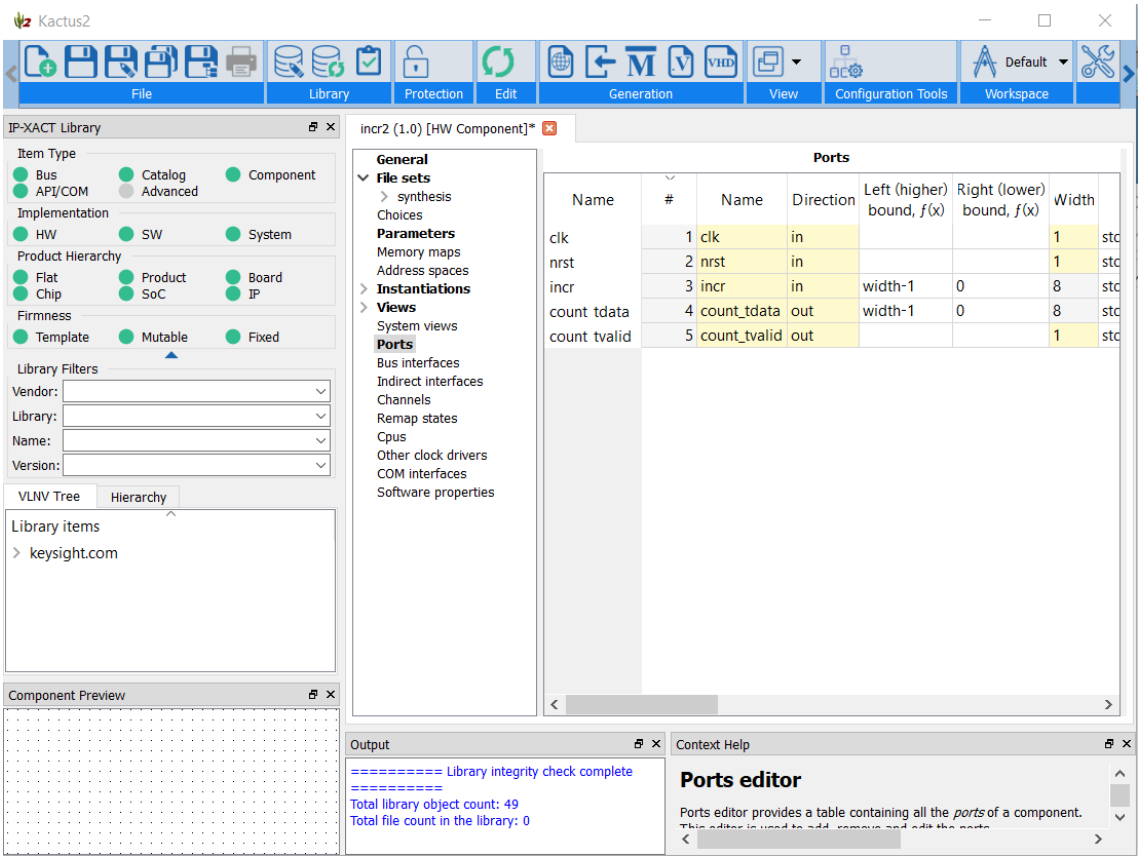
By default, Kactus2 includes all the files in the source directory. In the upper-right pane, click on **File sets/synthesis** to bring up the file set editor:



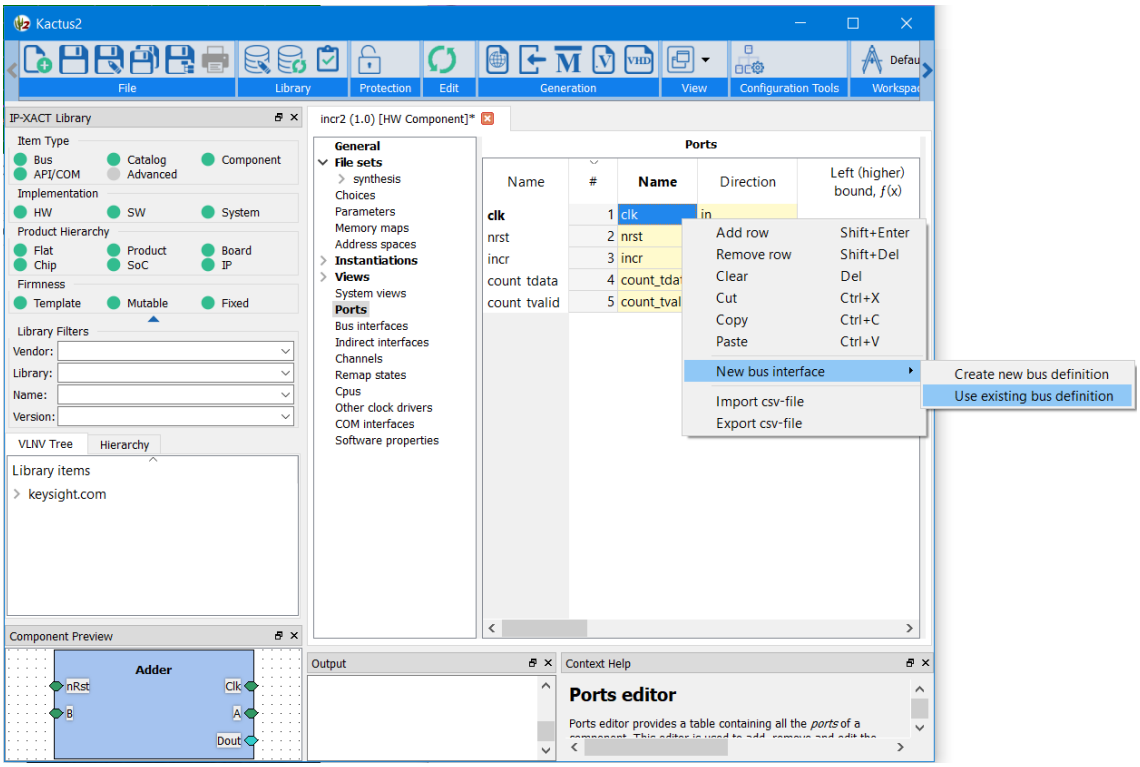
If the list of files contains files that are not part of desired IP blocks source, delete them using **right-click/remove row** or **Shift+del**. Note that in this case, there is only one source file, but for more elaborate designs there may be multiple files. If so, they should all be included. If necessary, they can be manually added if they are in a different directory.



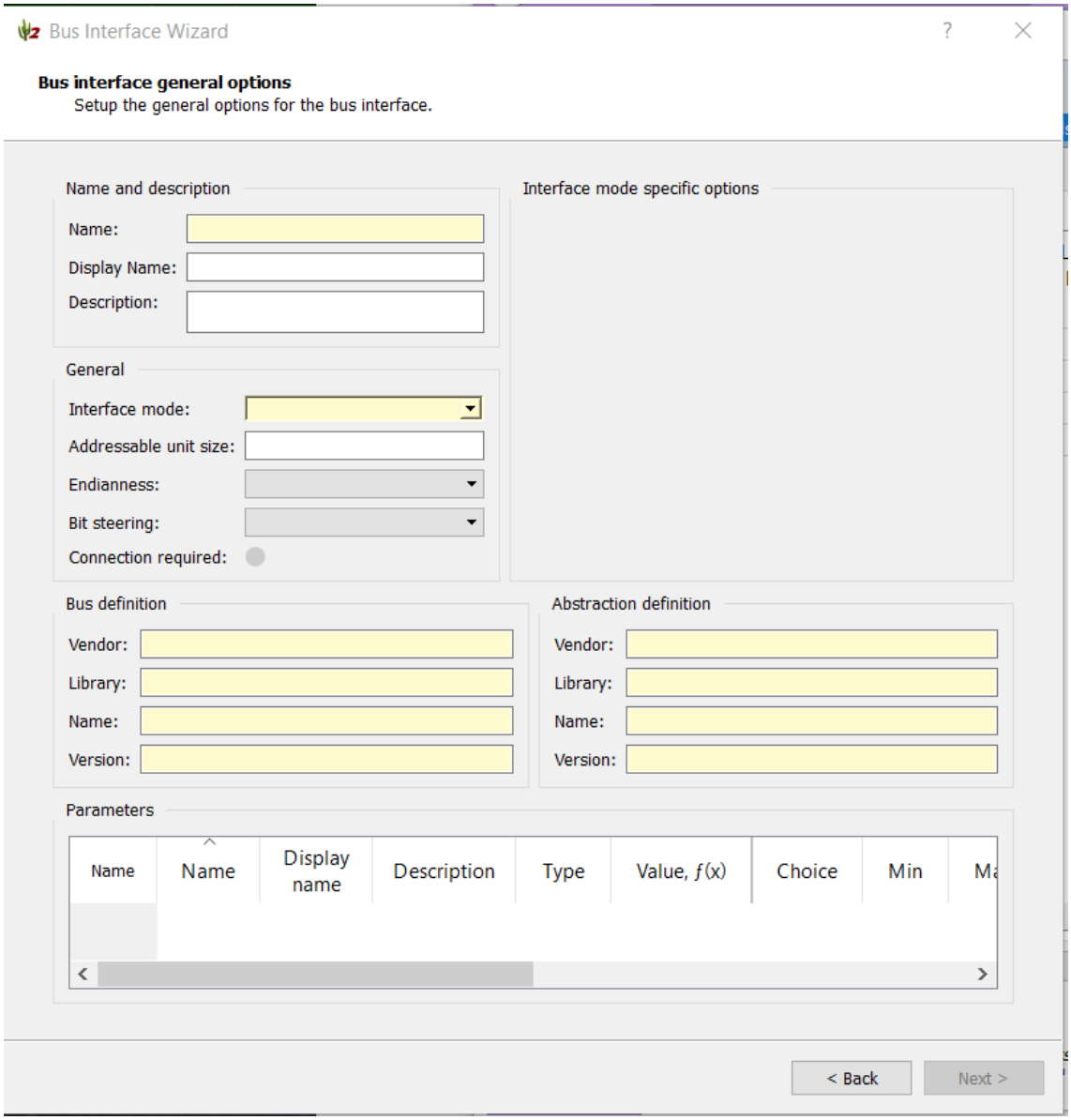
Now that the list of source files has been fixed, it is time to assign ports to logical *Bus Interfaces*. This allows PathWave FPGA to more easily connect interfaces between IP blocks. Click on **Ports** to bring up the Ports Editor. This should show all the ports that were read from the source HDL file and shows things like the direction (in or out), the width, and the index bounds for vectors. Note that in contrast with the example with the unparameterized IP block, in this case the higher bound of the *incr* and *count_tdata* ports show an expression involving the *width* parameter. In particular the upper bound is the expression *width-1*. Parameters can be used by themselves or in mathematical expressions as shown here. In this case, the default value of *width* is 8, and since that hasn't been changed, the entries in the Width column for these two ports shows the value 8. If the *width* parameter is changed, these fields will update with the new value. Further, if you do a mouse-over by placing the cursor over the expression *width-1*, you will see that the current value of the expression, 7, is shown.



To assign a single port to an interface, select the port in the yellow box under the **Name** column, right-click and select **New bus interface/Use existing bus definition**.



This will bring up the **Bus Interface Wizard**. This wizard will be used to assign all the ports to interfaces:

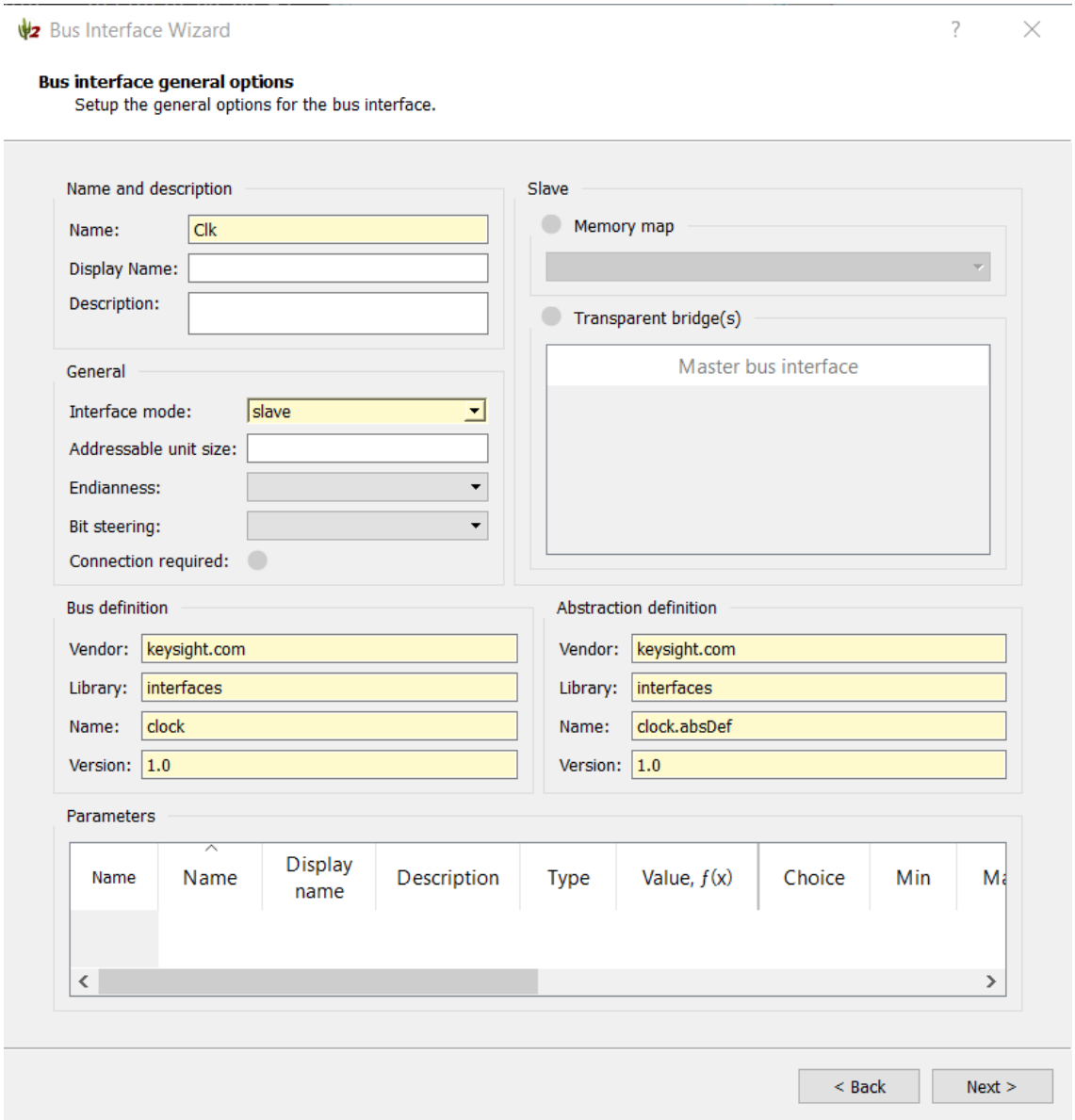


At the **Introduction** screen, click **Next** to bring up the **Bus interface general options** page.

Note that the boxes shaded in yellow are required fields that need to be filled out. White boxes are optional fields. Select the **Name:** box and enter a name for the interface. This is typically the name of the port, though it does not have to be. Next the **Bus definition** and **Abstraction definition** fields need to be filled out. Data entry can be speeded up by using the tab key.

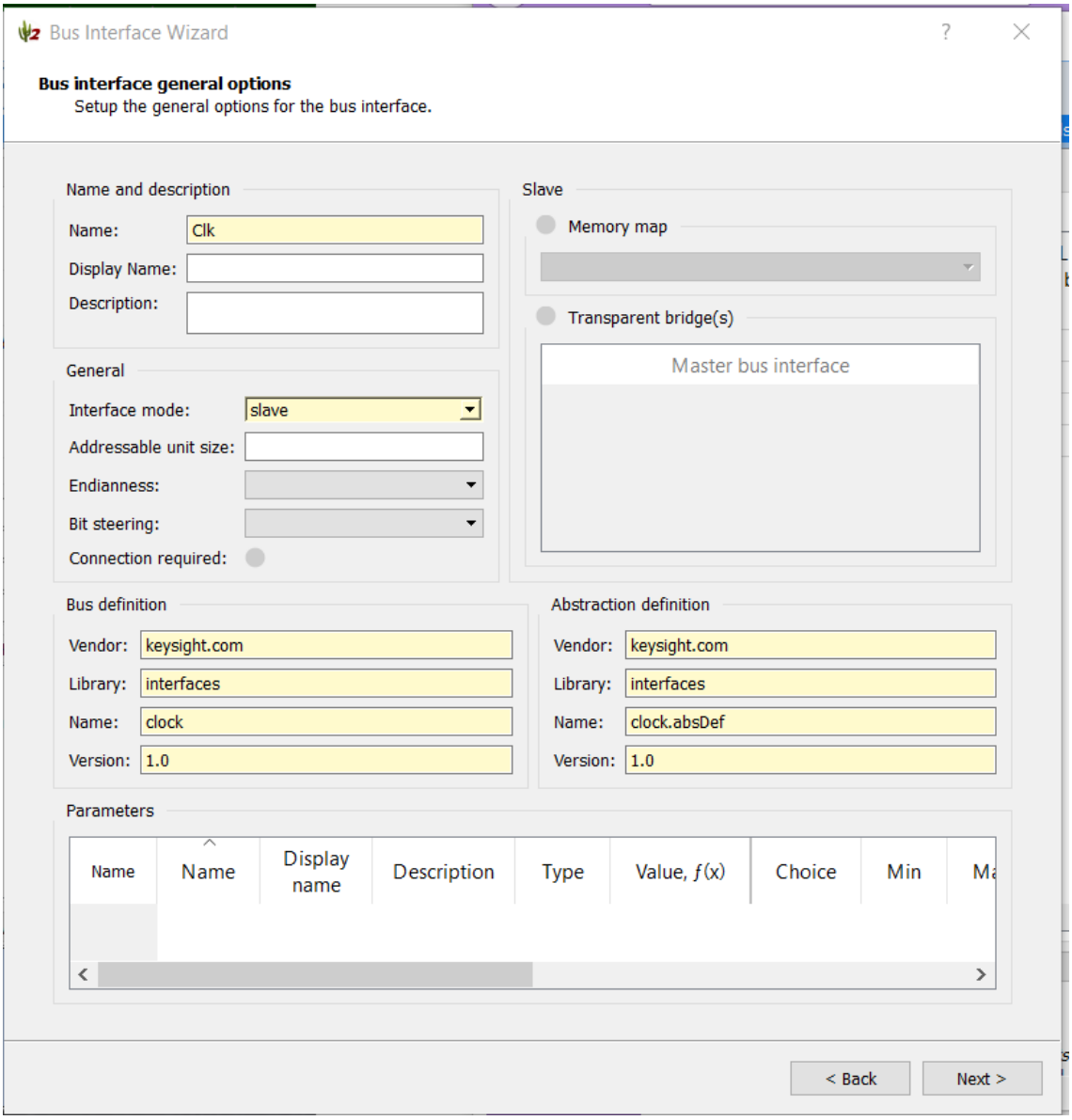
Click on **Vendor:** and *keysight.com* should show up as a suggested entry. Press the tab key to select *keysight.com* and move on to the next field, **Library**. Here, *interfaces* should show up as a suggested entry. Press the tab key to select *interfaces* and move on to the **Name:** field. Alternately, click on the *interfaces* entry to select it and then click on the **Name:** entry to advance to that field.

Under **Name:** select the type of interface that this port should be assigned to. In this case the port is a clock signal, so *clock* should be picked:

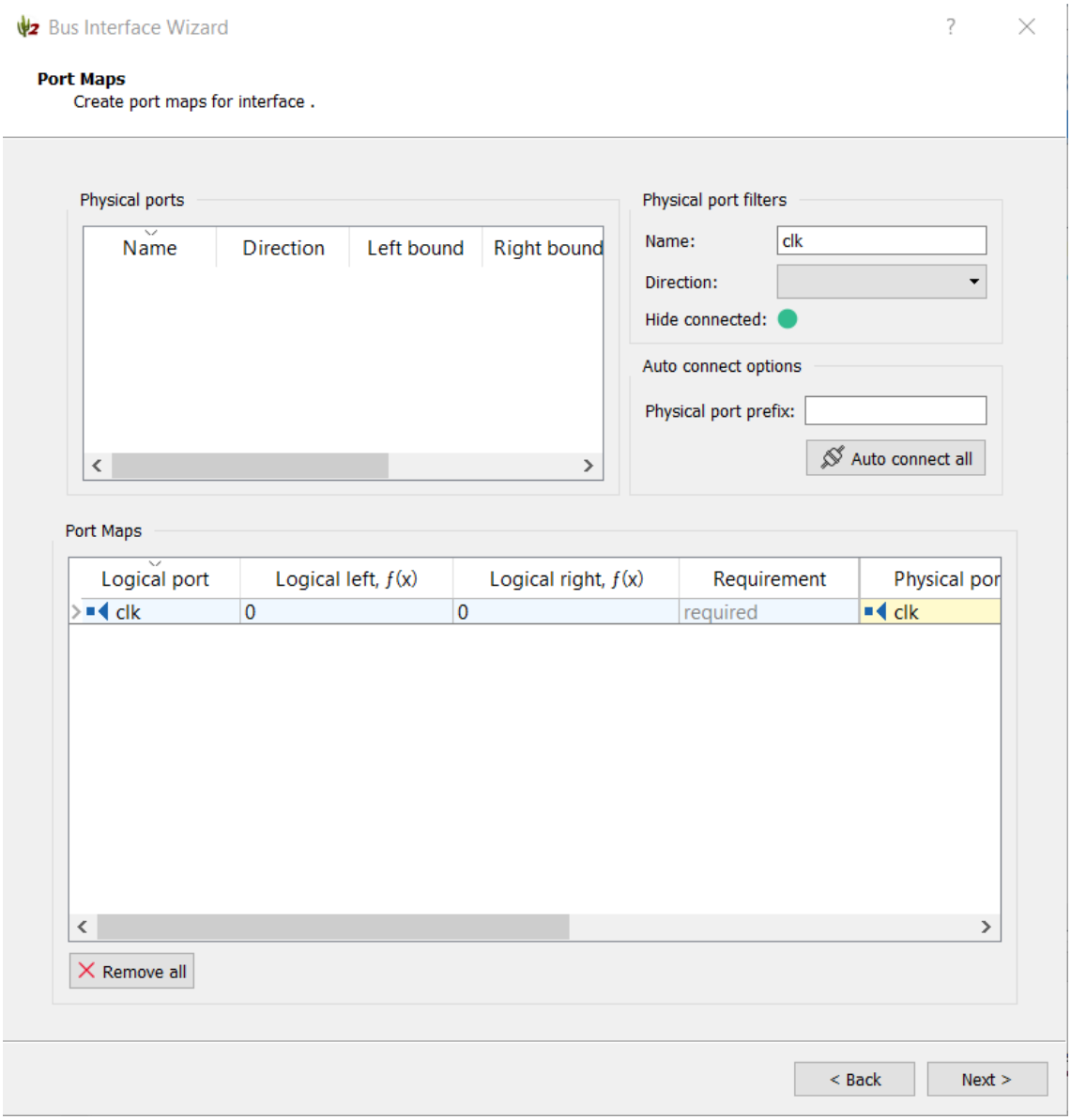


Note that one can speed up the selection by starting to type the name *cl...* to skip down the list more quickly.

After selecting *clock*, press the tab key to select the **Version:**. Press tab four more times to fill out the default **Abstraction definition** fields. The last tab will place the cursor in the **Interface mode:** field. This selects whether the interface is a *master*, meaning it generates the signal, or a *slave*, meaning it consumes the signal. In this case, the clock port is an input port and hence *slave* should be selected:

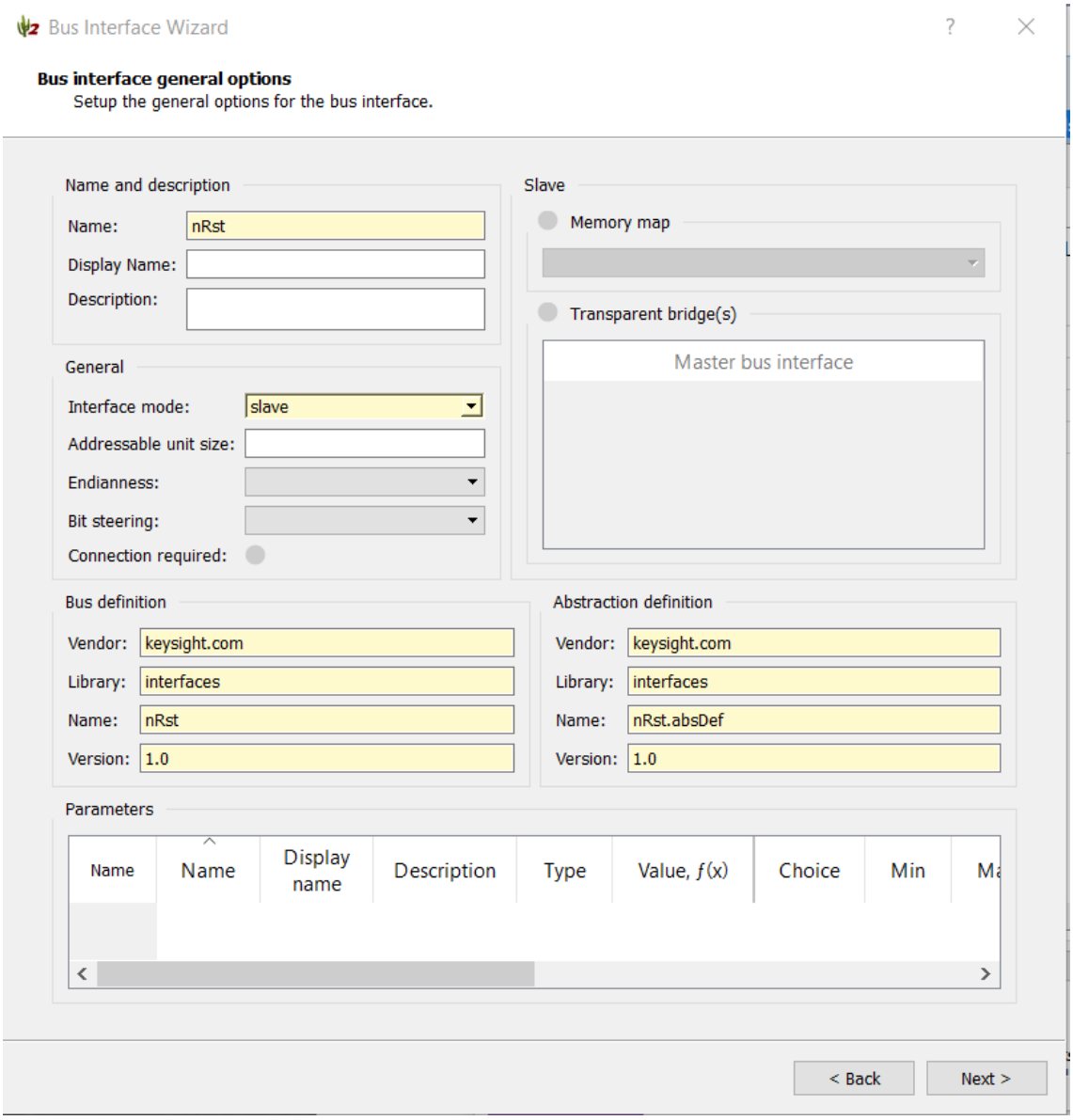


Next the ports need to be assigned to their various roles within the interface. For interfaces with only one port, this is trivial. Click **Next** twice to get to the **Port Maps**. Here the port mapping would be set, but in this case there is only one port so it is automatically filled in:

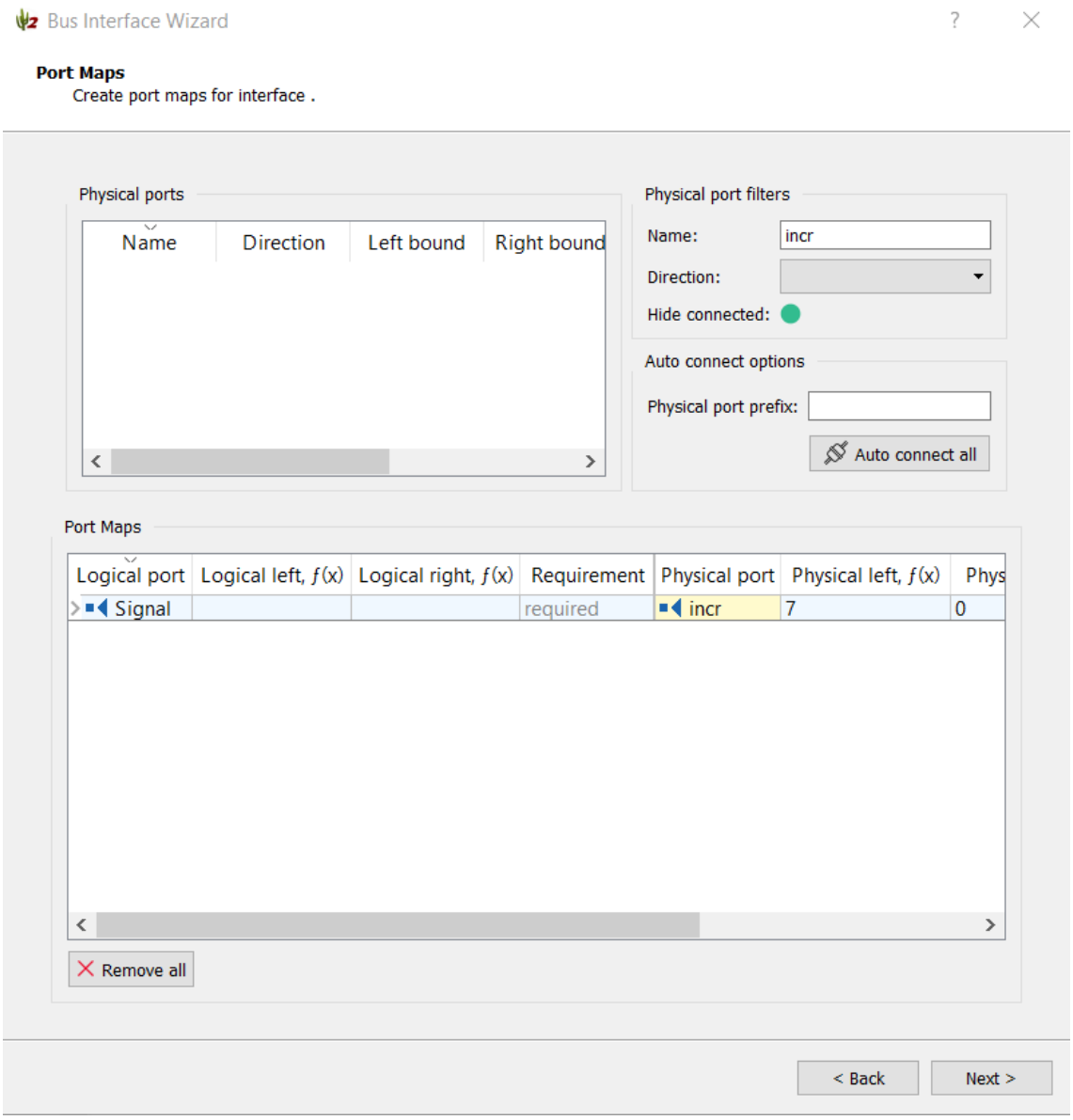


Click **Next** and **Finish** to complete the definition of this interface.

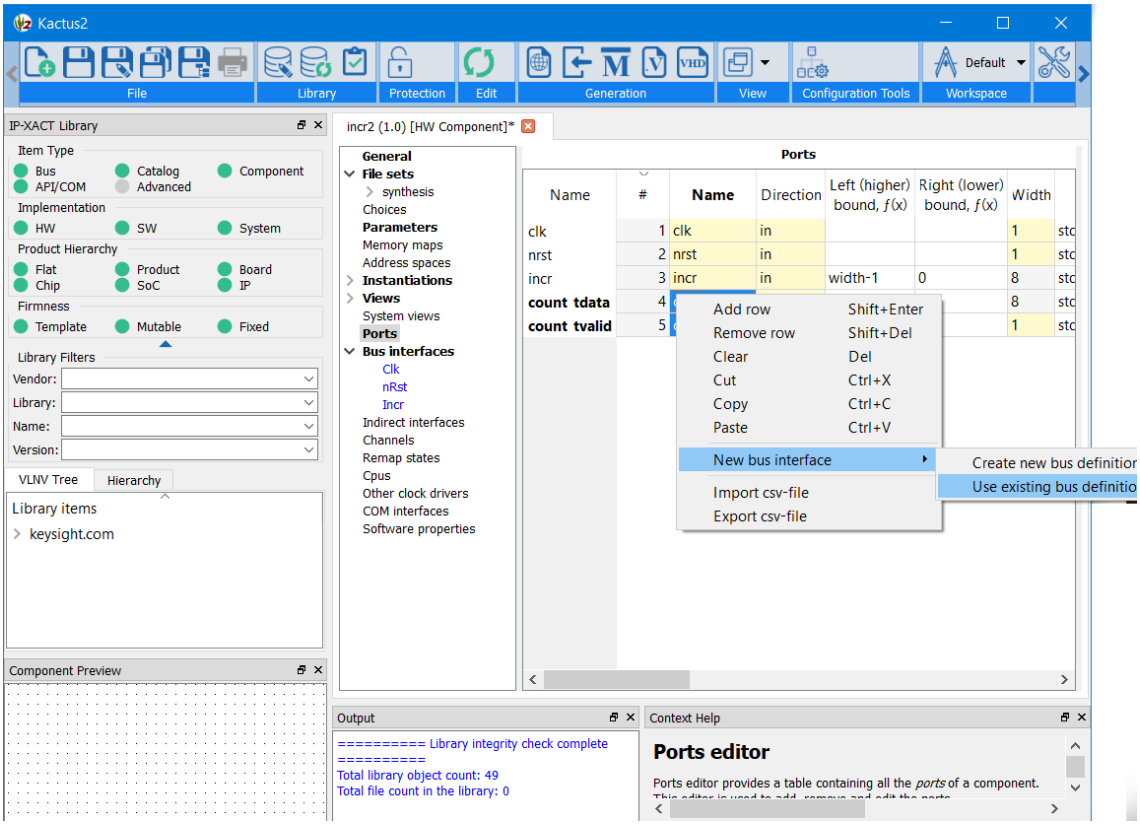
Repeat this process with the *nrst* port using the *nRst* interface:



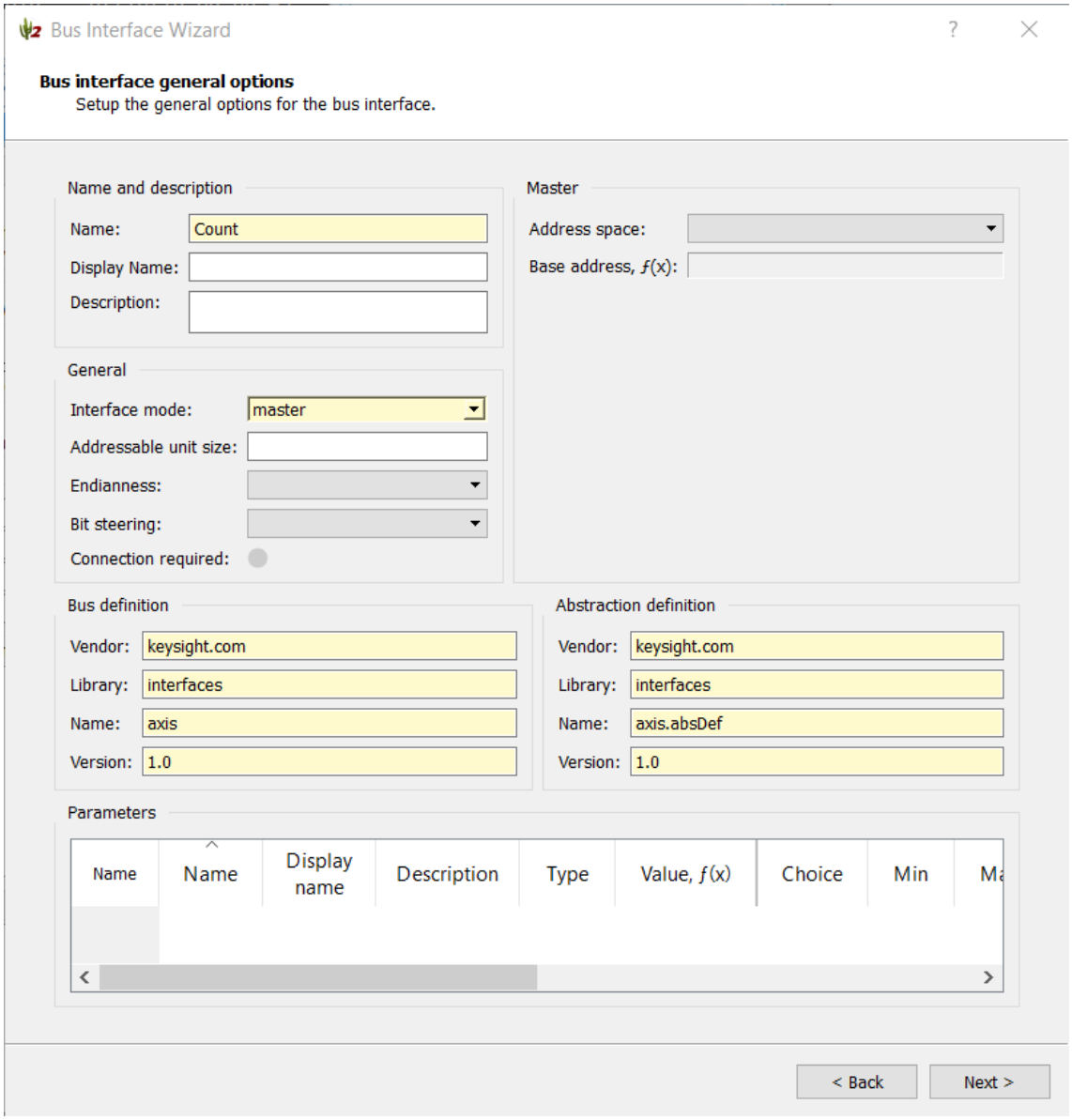
This process works for logic vectors too. Select the *incr* port and configure it as a *vector* interface:



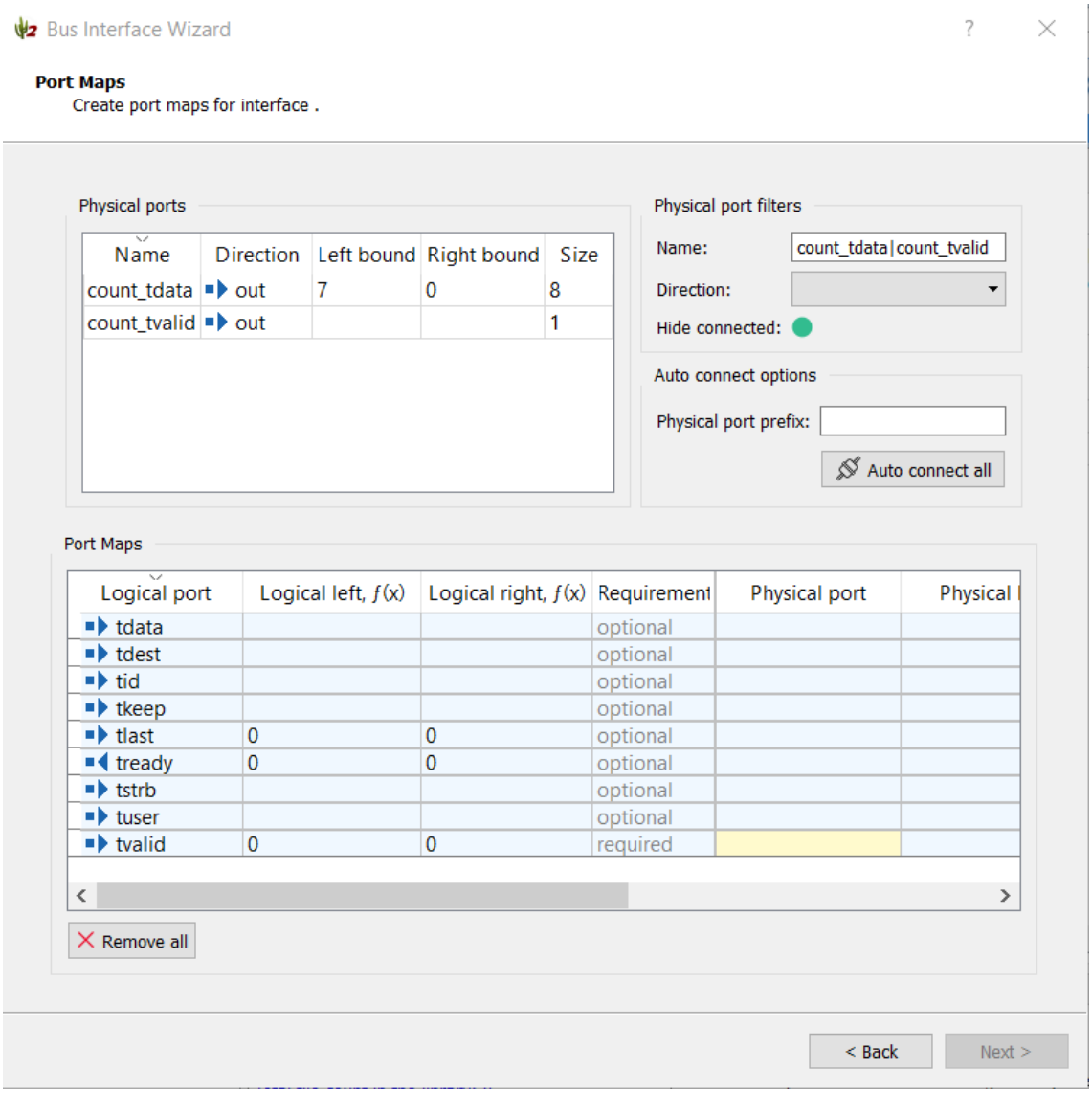
The interface for the *count* ports is a little different. In this case there is more than one port associated with the one interface. This is a case of an AXI-streaming interface consisting of both the *count_tdata* and *count_tvalid* ports. Select both the *count_tdata* and *count_tvalid* ports, right-click, and select the **New bus interface/Use existing bus definition** as before (note: it is okay to select more than the ports associated with the interface as long as all the ports that are associated with the interface are selected):



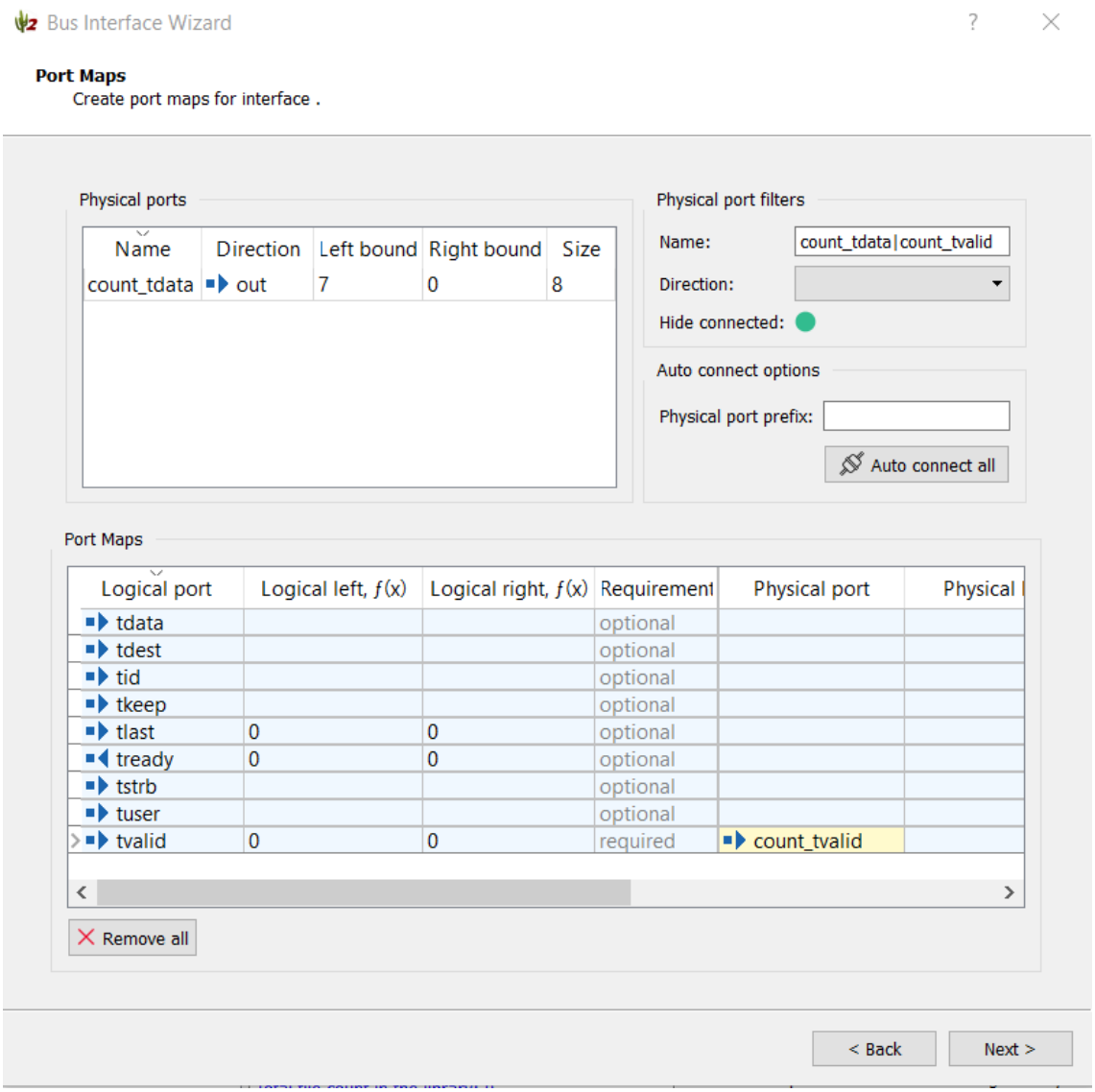
Fill in the **Bus interface general options** page using the *axis* interface name. In this case, the ports are outputs and the interface is a *master*:



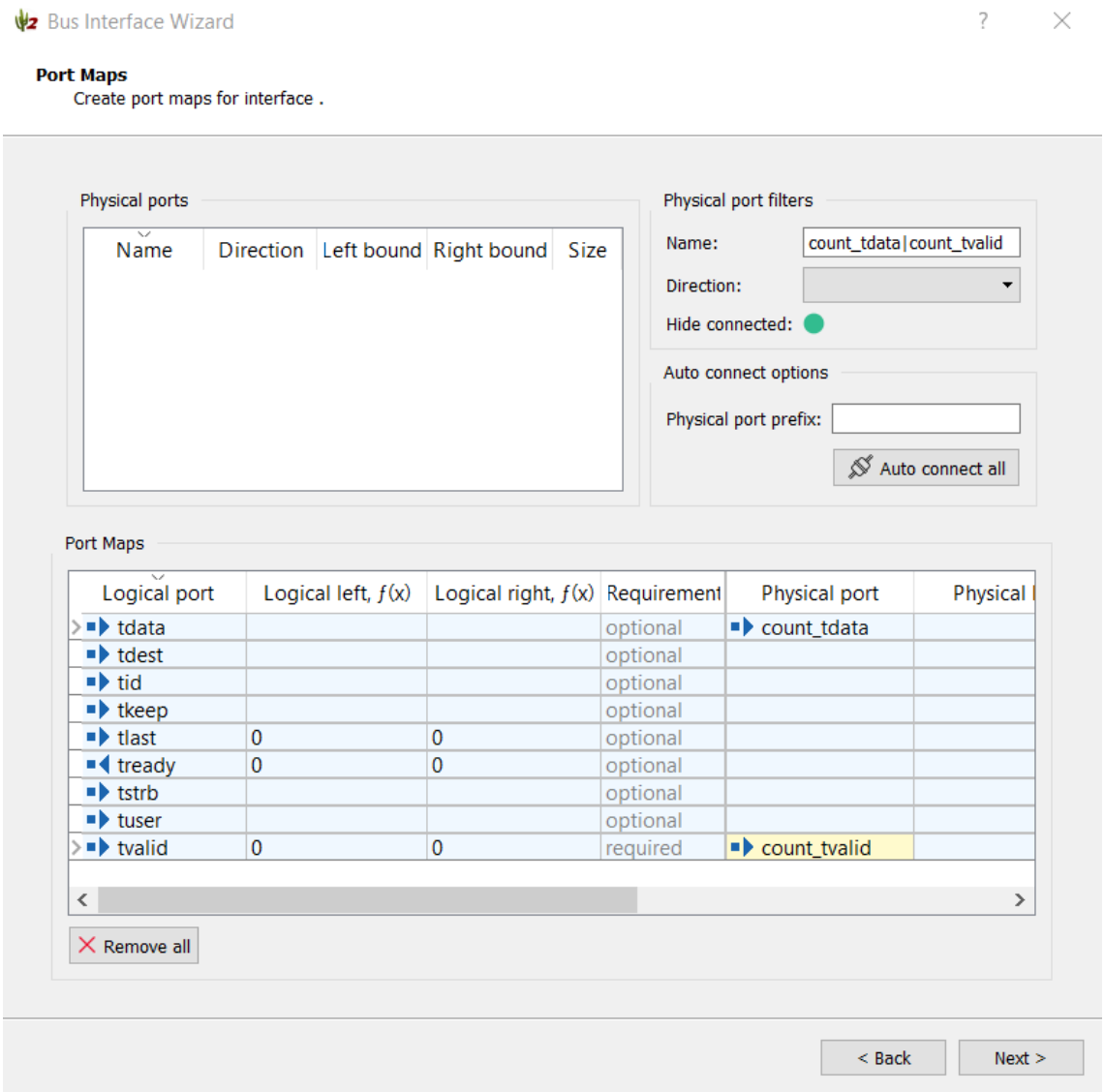
Now at the **Port Maps** page, one sees that there are multiple logical ports listed. Note that only the *tvalid* line has a yellow tinted box. That is the only port required by the AXI streaming spec. All the other ports are optional. Of these, this IP block only uses the *tdata* logical port.



To assign the *count_tvalid* physical port to the *tvalid* logical port, select *count_valid* and drag it down to the *tvalid* row:

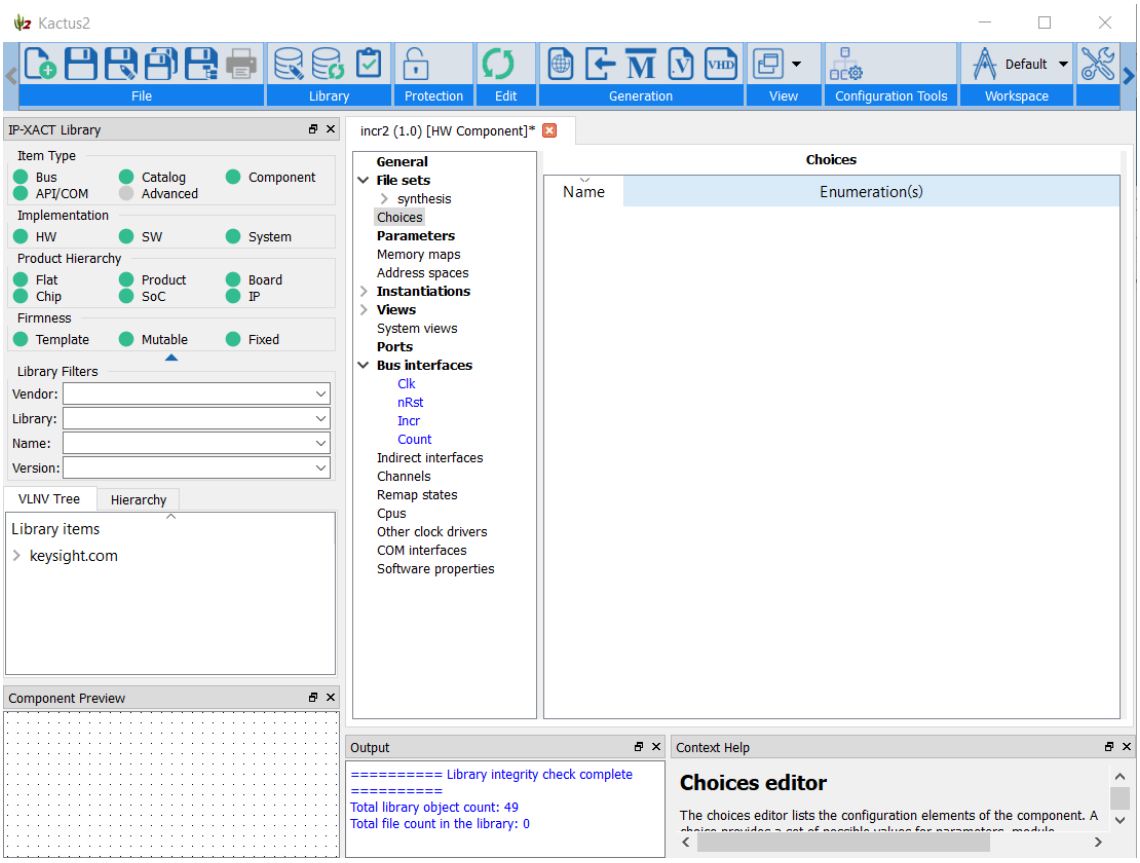


Likewise, drag the *count_tdata* physical port down to the *tdata* line:

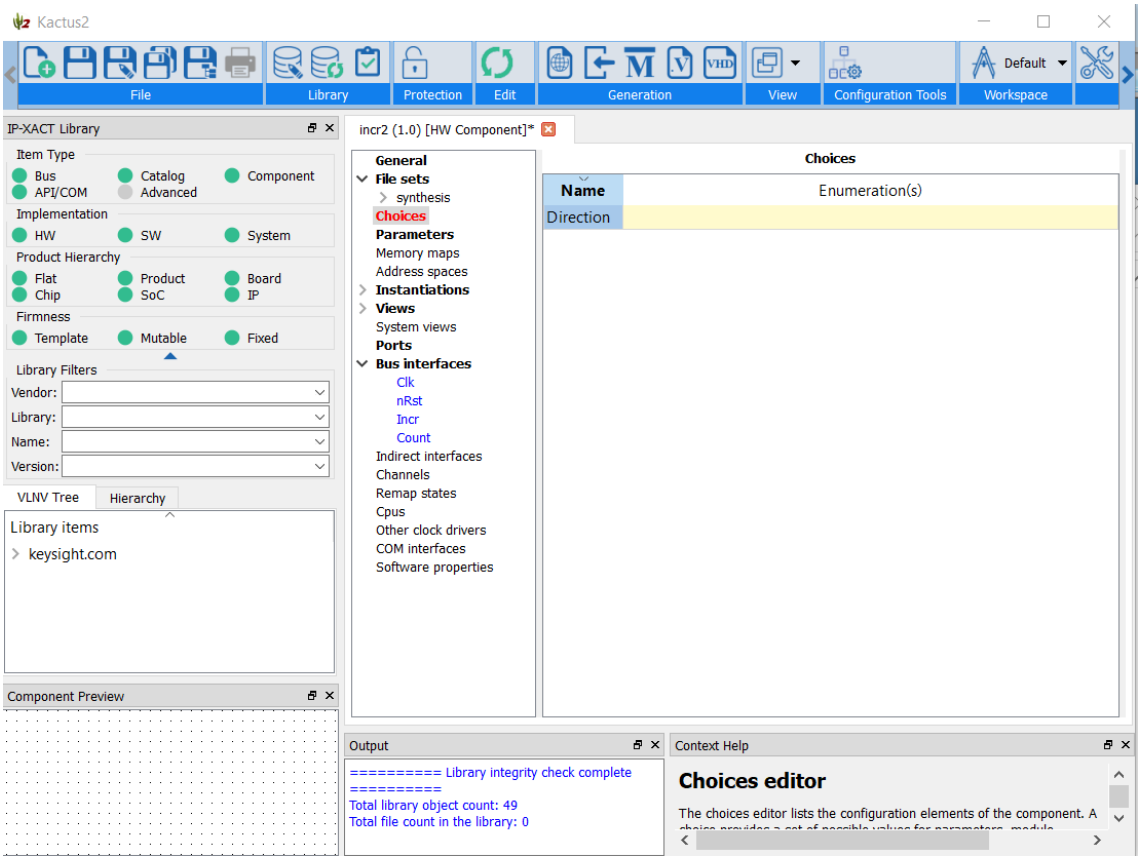


Click **Next** and **Finish** to complete the interface. At this point one can see that under the **Bus interfaces** section, all four of the IP block's interfaces are now listed.

So far, these steps have been the same as in the non-parameterized tutorial. Now it is time to work on the parameters. The *dir* parameter indicates the direction of the counting, up or down. Using the value 0 for up and 1 for down isn't very intuitive. Instead of using these integer values, an enumeration can be used to restrict the choices to a set of values and these values have names that can be more informative. In IP-XACT, enumerations are called *choices*. Click on the **Choices** entry in the center pane to bring up that window:

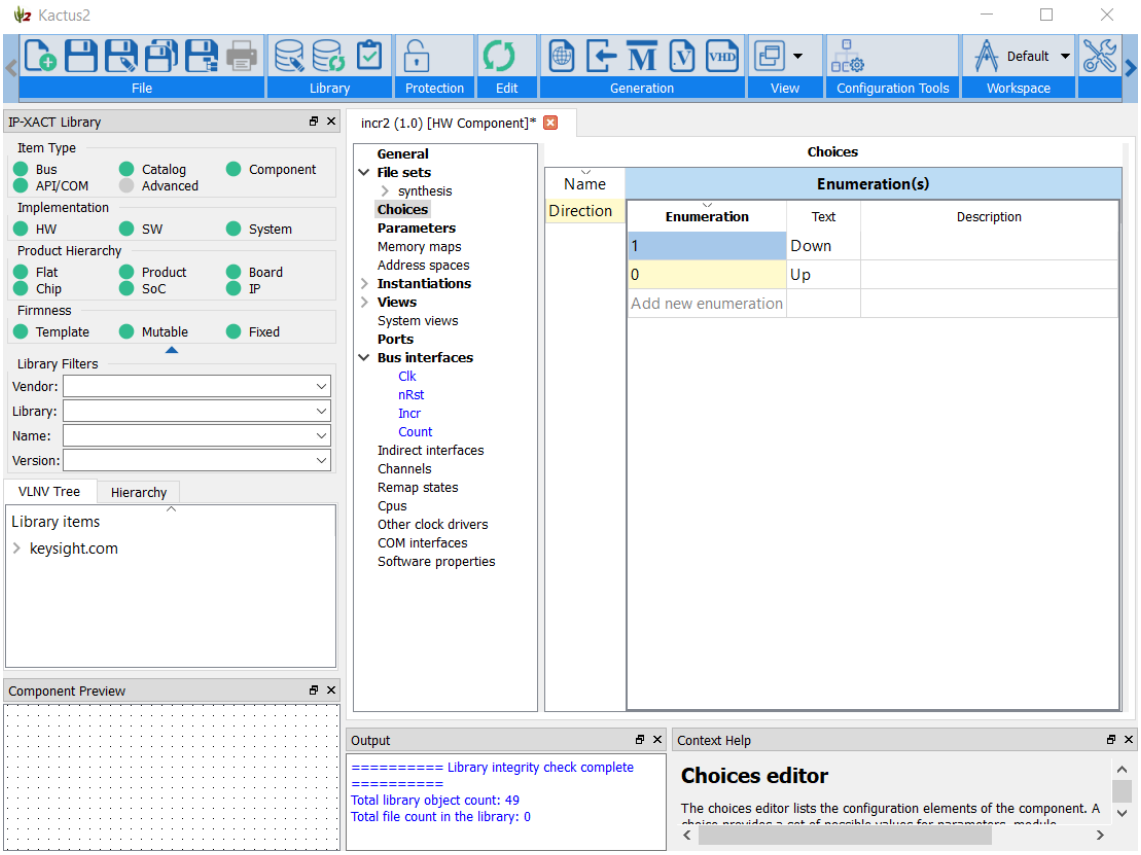


Double click in the **Choices** pane to create a new choice, click in the **Name** field to select it and enter the name **Direction**:

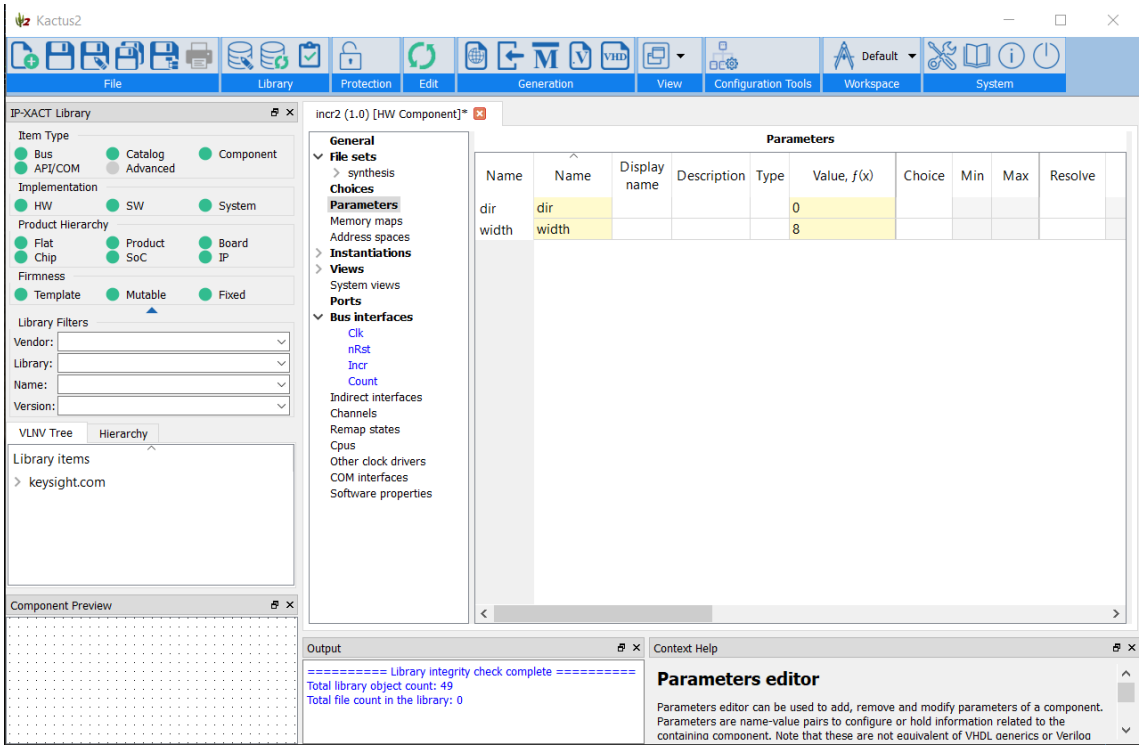


Notice that currently the **Choices** entry in the middle pane is red, indicating an error condition. That is because there is a choice (named **Direction**) defined, but the possible values of **Direction** have not yet been specified.

Double click in the **Enumeration(s)** entry to bring up the Enumeration Entry window. Enter two values, 0 with the text label **Up**, and 1 with the text label **Down**:



Now that the Choice has been specified, it can be used to describe a parameter. Click on the **Parameters** entry in the center pane to bring up that window. There are two parameters, *dir* and *width* with the default values of 0 and 8 respectively:



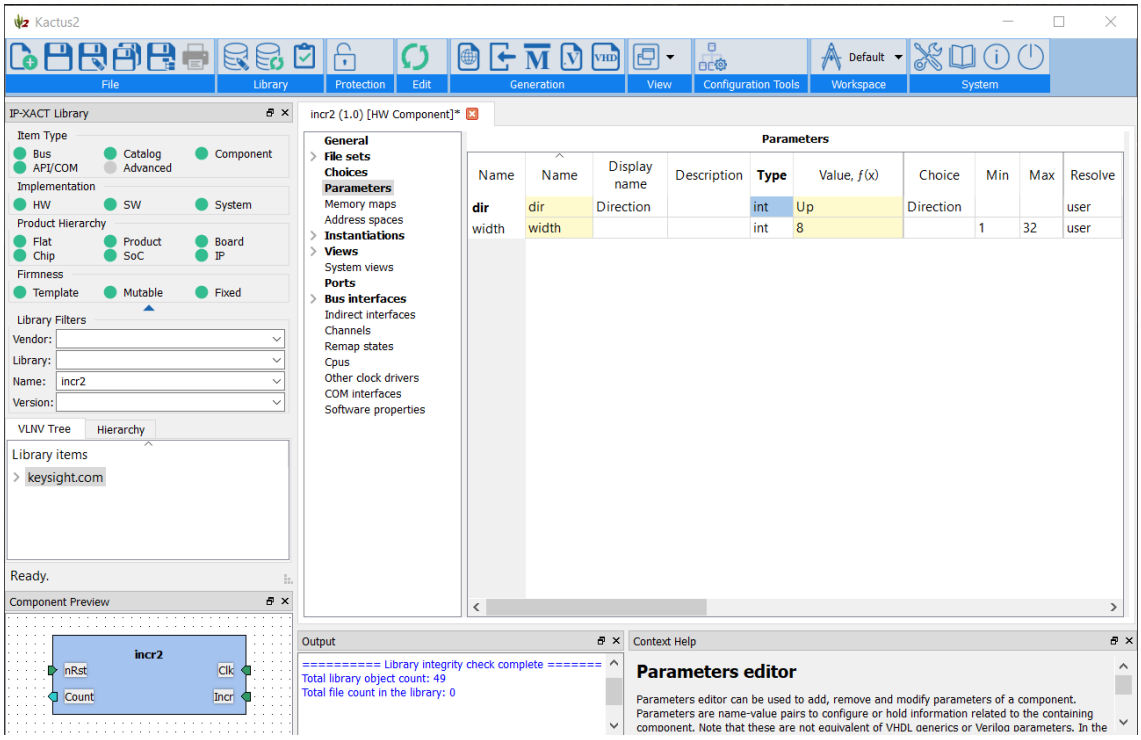
Change *dir* to use this Choice by clicking in the **Choice** column for *dir* and selecting the name of the choice just created *Direction*.

Since "dir" isn't that friendly of a name for the end user, put "Direction" in for the **Display name**. This is the text that will be presented to the end user when the IP block is used and customized.

Since both of these are parameters that the user should be given the choice of changing, the **Resolve** field of both should be set to *user*. Other values of the **Resolve** field can indicate parameters that are either calculated from other parameters or parameters that the user should not change.

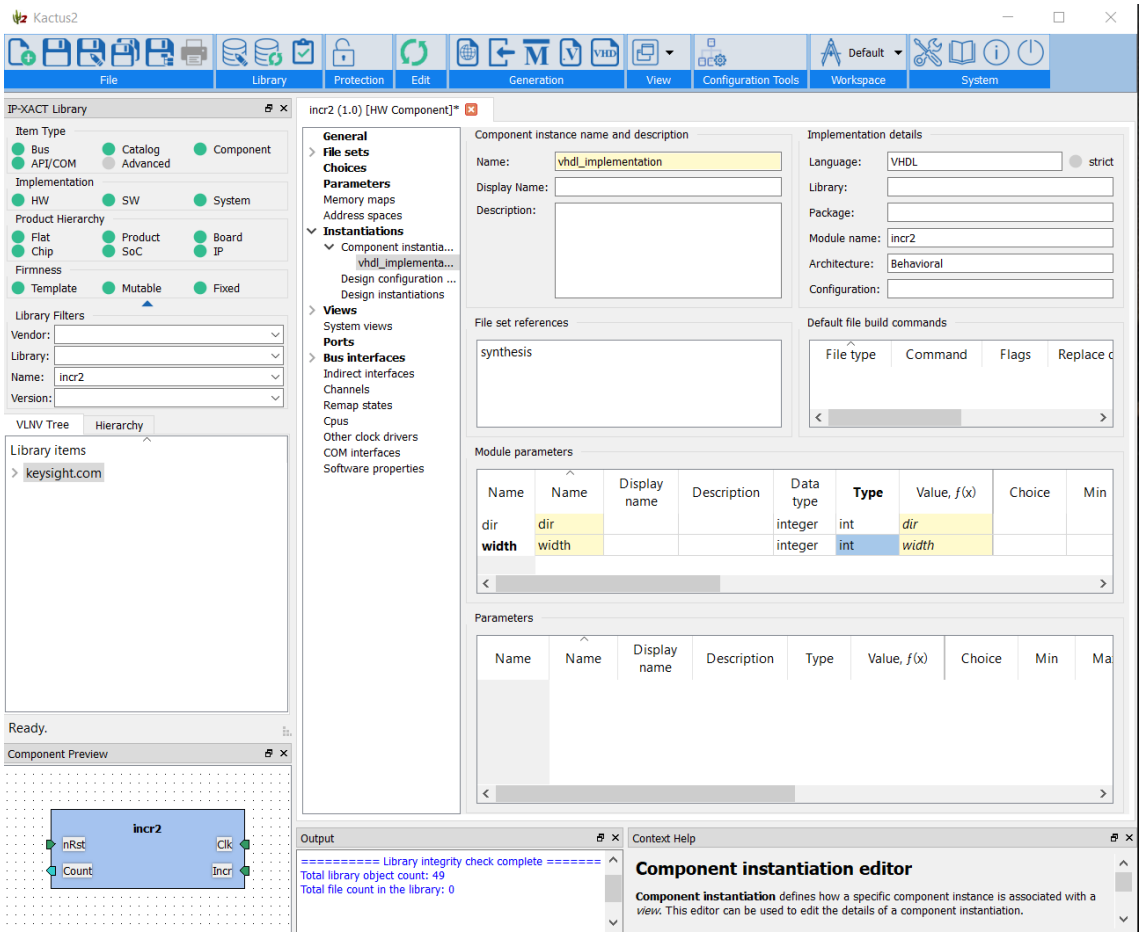
The **Type** of the parameter needs to be specified. In this case, they are both *int*s. Set these using the pull down selections in the **Type** column.

Limits can be set to restrict the allowable values of a parameter. In this example, the *width* parameter is restricted to a minimum of 1 and a maximum of 32.



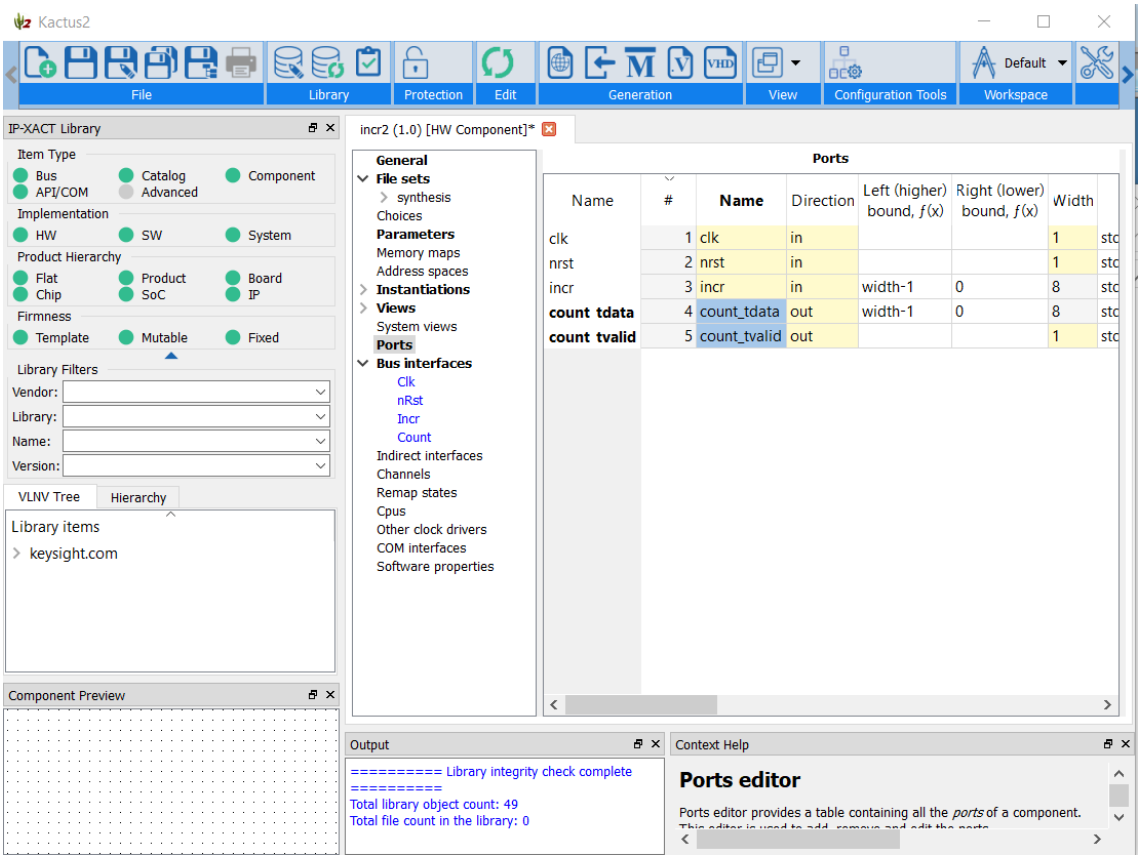
The parameters just defined are IP-XACT parameters and are used inside IP-XACT. The actual HDL generics that are passed on to the IP HDL are known as *moduleParameters*. Normally these will have the same names as the IP-XACT parameters and will just be copies. This is automatically done by Kactus2. To see this, click on the **Instantiations/Component instantiations/vhdl_implementation** in the center pane. In the *Module parameters* section of the right pane the module parameters are shown. The "dir" in the **Name** column refers to the module parameter by that name. The "*dir*" in the **Value** column refers to the IP-XACT parameter by that name.

The **Type** of these parameters must be set just as the IP-XACT parameters were set. In the same way as the previous screen, use the pull down selections in the **Type** column of the **Module parameters** window to change the type to *int*s.



The definition of the IP block's interfaces and parameters is complete. If any of the entries in the middle pane are red (none are in this case), that would indicate that there is an error with that entry. Select it and fix any errors until none of the entries are red.

Click the Save icon (or type Ctrl+S) to save the IP-XACT file.



If you are using Kactus2 version 3.5.77 or later, the next step can be skipped. For earlier versions of Kactus2, the following step is required.

When generating IP-XACT for parameterized IP blocks with Kactus2, there is one further step required. Kactus2 adds a **usageCount** field to the parameter definition. This field is not in the IP-XACT spec and is not valid IP-XACT XML. These fields need to be removed manually in a regular text editor. Edit the file *incr2.1.0.xml* that Kactus2 just generated. Search for **usageCount**:

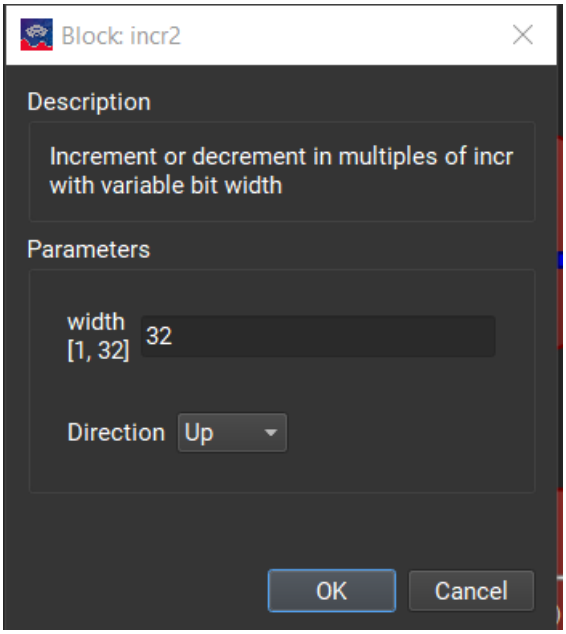
```
      <ipxact:file>
        <ipxact:name>src/incr2.vhd</ipxact:name>
        <ipxact:fileType>vhdlSource</ipxact:fileType>
        <ipxact:vendorExtensions>
          <kactus2:hash>80784e10e1b2a7e3a70fb0592f377473649faa02</kactus2:hash>
        </ipxact:vendorExtensions>
      </ipxact:file>
    </ipxact:fileSet>
  </ipxact:fileSets>
  <ipxact:description>Increment or decrement in multiples of incr with variable bit width</ipxact:description>
  <ipxact:parameters>
    <ipxact:parameter maximum="32" minimum="1" parameterId="uuid_69fca058_a0fb_4e1d_9db2_a713c9781c00" resolve="user" type="int" usageCount="3">
      <ipxact:name>width</ipxact:name>
      <ipxact:value>8</ipxact:value>
    </ipxact:parameter>
    <ipxact:parameter choiceRef="Direction" parameterId="uuid_74620f8f_d6ae_4da4_b710_5c6c86e460cf" resolve="user" usageCount="1">
      <ipxact:name>dir</ipxact:name>
      <ipxact:displayName>Direction</ipxact:displayName>
      <ipxact:value>0</ipxact:value>
    </ipxact:parameter>
  </ipxact:parameters>
  <ipxact:vendorExtensions>
```

and delete that field:

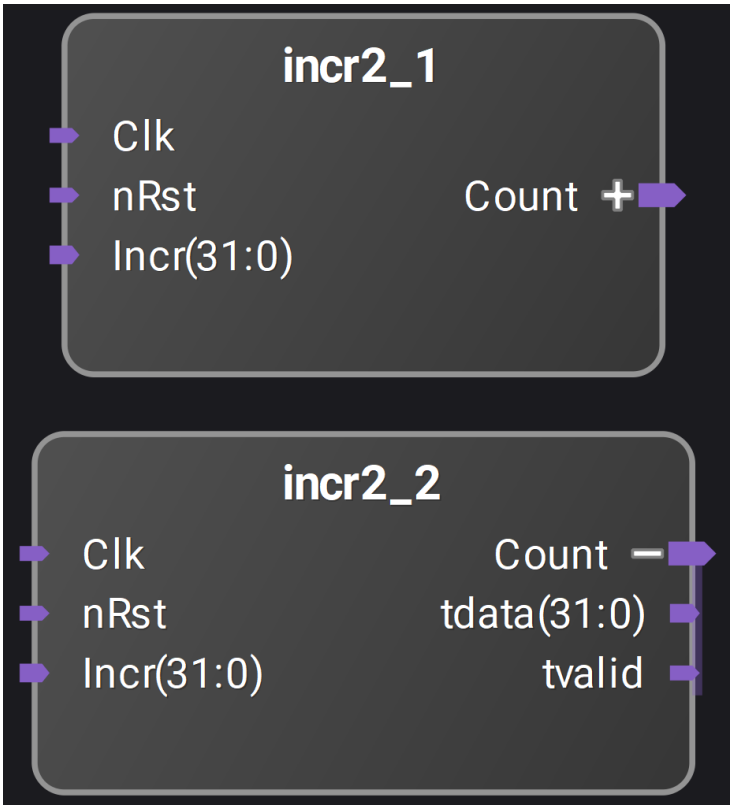
```
      <ipxact:name>src/incr2.vhd</ipxact:name>
      <ipxact:fileType>vhdlSource</ipxact:fileType>
      <ipxact:vendorExtensions>
        <kactus2:hash>80784e10e1b2a7e3a70fb0592f377473649faa02</kactus2:hash>
      </ipxact:vendorExtensions>
    </ipxact:file>
  </ipxact:fileSet>
</ipxact:fileSets>
<ipxact:description>Increment or decrement in multiples of incr with variable bit width</ipxact:description>
<ipxact:parameters>
  <ipxact:parameter maximum="32" minimum="1" parameterId="uuid_69fca058_a0fb_4e1d_9db2_a713c9781c00" resolve="user" type="int">
    <ipxact:name>width</ipxact:name>
    <ipxact:value>8</ipxact:value>
  </ipxact:parameter>
  <ipxact:parameter choiceRef="Direction" parameterId="uuid_74620f8f_d6ae_4da4_b710_5c6c86e460cf" resolve="user">
    <ipxact:name>dir</ipxact:name>
    <ipxact:displayName>Direction</ipxact:displayName>
    <ipxact:value>0</ipxact:value>
  </ipxact:parameter>
</ipxact:parameters>
```

and save the file.

The description of this block's interfaces is now complete and PathWave FPGA can now use this interface information to allow easier connections to other blocks. When that block is used within PathWave FPGA, the following dialog box will show up. This shows the description of the IP block along with the user modifiable parameters. In this case there are the two parameters we defined above: **width** with a valid range of [1,32], and the enumeration for **Direction** with the choices **Up** and **Down**.



In this example, the user changed the *width* parameter from the default value of 8 to the value 32. This causes the I/O ports to have a range of (31:0) rather than (7:0).



In this screen capture from PathWave FPGA, the instance incr2_1 is shown with the *Count* interface collapsed. The internal ports that make up that interface are not shown and the interface can be connected to other compatible interfaces with one connection. The instance incr2_2 is shown with the *Count* interface expanded to show the internal ports that make up that interface. The entire interface can be connected with one connection by using the *Count* port or the individual ports within the interface can be connected separately if desired.

The generated IP-XACT is

Code Block 9 incr2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ipxact:component xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014"
xmlns:kactus2="http://kactus2.cs.tut.fi"
xsi:schemaLocation="http://www.accellera.org/XMLSchema/IPXACT/1685-2014
http://www.accellera.org/XMLSchema/IPXACT/1685-2014/index.xsd">
 <ipxact:vendor>keysight.com</ipxact:vendor>
 <ipxact:library>flat</ipxact:library>
 <ipxact:name>incr2</ipxact:name>
 <ipxact:version>1.0</ipxact:version>
 <ipxact:busInterfaces>
    <ipxact:busInterface>
       <ipxact:name>Clk</ipxact:name>
       <ipxact:busType vendor="keysight.com" library="interfaces"
name="clock" version="1.0"/>
       <ipxact:abstractionTypes>
          <ipxact:abstractionType>
             <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="clock.absDef" version="1.0"/>
             <ipxact:portMaps>
                <ipxact:portMap>
                   <ipxact:logicalPort>
                      <ipxact:name>clk</ipxact:name>
                      <ipxact:range>
                         <ipxact:left>0</ipxact:left>
```

```
                        <ipxact:right>0</ipxact:right>
                     </ipxact:range>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>clk</ipxact:name>
                     <ipxact:partSelect>
                        <ipxact:range>
                           <ipxact:left>0</ipxact:left>
                           <ipxact:right>0</ipxact:right>
                        </ipxact:range>
                     </ipxact:partSelect>
                  </ipxact:physicalPort>
               </ipxact:portMap>
            </ipxact:portMaps>
         </ipxact:abstractionType>
      </ipxact:abstractionTypes>
      <ipxact:slave/>
   </ipxact:busInterface>
   <ipxact:busInterface>
      <ipxact:name>nRst</ipxact:name>
      <ipxact:busType vendor="keysight.com" library="interfaces"
name="nRst" version="1.0"/>
      <ipxact:abstractionTypes>
         <ipxact:abstractionType>
            <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="nRst.absDef" version="1.0"/>
            <ipxact:portMaps>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>nRst</ipxact:name>
                     <ipxact:range>
                        <ipxact:left>0</ipxact:left>
                        <ipxact:right>0</ipxact:right>
                     </ipxact:range>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>nrst</ipxact:name>
                     <ipxact:partSelect>
                        <ipxact:range>
                           <ipxact:left>0</ipxact:left>
                           <ipxact:right>0</ipxact:right>
                        </ipxact:range>
                     </ipxact:partSelect>
                  </ipxact:physicalPort>
               </ipxact:portMap>
            </ipxact:portMaps>
         </ipxact:abstractionType>
      </ipxact:abstractionTypes>
      <ipxact:slave/>
   </ipxact:busInterface>
   <ipxact:busInterface>
      <ipxact:name>Incr</ipxact:name>
      <ipxact:busType vendor="keysight.com" library="interfaces"
name="vector" version="1.0"/>
      <ipxact:abstractionTypes>
         <ipxact:abstractionType>
            <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="vector.absDef" version="1.0"/>
            <ipxact:portMaps>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>Signal</ipxact:name>
                     <ipxact:range>
                        <ipxact:left>7</ipxact:left>
                        <ipxact:right>0</ipxact:right>
                     </ipxact:range>
```

```
                </ipxact:logicalPort>
                <ipxact:physicalPort>
                   <ipxact:name>incr</ipxact:name>
                   <ipxact:partSelect>
                      <ipxact:range>
                         <ipxact:left>7</ipxact:left>
                         <ipxact:right>0</ipxact:right>
                      </ipxact:range>
                   </ipxact:partSelect>
                </ipxact:physicalPort>
             </ipxact:portMap>
          </ipxact:portMaps>
       </ipxact:abstractionType>
    </ipxact:abstractionTypes>
    <ipxact:slave/>
 </ipxact:busInterface>
 <ipxact:busInterface>
    <ipxact:name>Count</ipxact:name>
    <ipxact:busType vendor="keysight.com" library="interfaces"
name="axis" version="1.0"/>
    <ipxact:abstractionTypes>
       <ipxact:abstractionType>
          <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="axis.absDef" version="1.0"/>
          <ipxact:portMaps>
             <ipxact:portMap>
                <ipxact:logicalPort>
                   <ipxact:name>tvalid</ipxact:name>
                </ipxact:logicalPort>
                <ipxact:physicalPort>
                   <ipxact:name>count_tvalid</ipxact:name>
                </ipxact:physicalPort>
             </ipxact:portMap>
             <ipxact:portMap>
                <ipxact:logicalPort>
                   <ipxact:name>tdata</ipxact:name>
                </ipxact:logicalPort>
                <ipxact:physicalPort>
                   <ipxact:name>count_tdata</ipxact:name>
                </ipxact:physicalPort>
             </ipxact:portMap>
          </ipxact:portMaps>
       </ipxact:abstractionType>
    </ipxact:abstractionTypes>
    <ipxact:master/>
 </ipxact:busInterface>
</ipxact:busInterfaces>
<ipxact:model>
 <ipxact:views>
    <ipxact:view>
       <ipxact:name>flat_vhdl</ipxact:name>
       <ipxact:envIdentifier>VHDL:Kactus2:</ipxact:envIdentifier>

<ipxact:componentInstantiationRef>vhdl_implementation</ipxact:componentI
nstantiationRef>
    </ipxact:view>
 </ipxact:views>
 <ipxact:instantiations>
    <ipxact:componentInstantiation>
       <ipxact:name>vhdl_implementation</ipxact:name>
       <ipxact:language>VHDL</ipxact:language>
       <ipxact:moduleName>incr2</ipxact:moduleName>
       <ipxact:architectureName>Behavioral</ipxact:architectureName>
       <ipxact:moduleParameters>
          <ipxact:moduleParameter dataType="integer"
parameterId="uuid_9b050478_c9d3_4507_8dc8_7ea7c47f93ac" type="int"
```

```
usageType="nontyped">
               <ipxact:name>width</ipxact:name>

 <ipxact:value>uuid_69fca058_a0fb_4e1d_9db2_a713c9781c00</ipxact:value>
            </ipxact:moduleParameter>
            <ipxact:moduleParameter dataType="integer"
parameterId="uuid_c6f0c841_b30e_4869_9529_173232f1d921" type="int"
usageType="nontyped">
               <ipxact:name>dir</ipxact:name>

 <ipxact:value>uuid_74620f8f_d6ae_4da4_b710_5c6c86e460cf</ipxact:value>
            </ipxact:moduleParameter>
         </ipxact:moduleParameters>
         <ipxact:fileSetRef>
            <ipxact:localName>synthesis</ipxact:localName>
         </ipxact:fileSetRef>
      </ipxact:componentInstantiation>
   </ipxact:instantiations>
   <ipxact:ports>
      <ipxact:port>
         <ipxact:name>clk</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>std_logic</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>nrst</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>std_logic</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>incr</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:vectors>
               <ipxact:vector>
                  <ipxact:left>uuid_69fca058_a0fb_4e1d_9db2_a713c9781c00-
1</ipxact:left>
                  <ipxact:right>0</ipxact:right>
               </ipxact:vector>
            </ipxact:vectors>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>std_logic_vector</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
```

```
         <ipxact:name>count_tdata</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>out</ipxact:direction>
            <ipxact:vectors>
               <ipxact:vector>
                  <ipxact:left>uuid_69fca058_a0fb_4e1d_9db2_a713c9781c00-
1</ipxact:left>
                  <ipxact:right>0</ipxact:right>
               </ipxact:vector>
            </ipxact:vectors>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>std_logic_vector</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>count_tvalid</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>out</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>std_logic</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
   </ipxact:ports>
 </ipxact:model>
 <ipxact:choices>
   <ipxact:choice>
      <ipxact:name>Direction</ipxact:name>
      <ipxact:enumeration text="Up">0</ipxact:enumeration>
      <ipxact:enumeration text="Down">1</ipxact:enumeration>
   </ipxact:choice>
 </ipxact:choices>
 <ipxact:fileSets>
   <ipxact:fileSet>
      <ipxact:name>synthesis</ipxact:name>
      <ipxact:file>
         <ipxact:name>src/incr2.vhd</ipxact:name>
         <ipxact:fileType>vhdlSource</ipxact:fileType>
         <ipxact:vendorExtensions>

 <kactus2:hash>80784e10e1b2a7e3a70fb0592f377473649faa02</kactus2:hash>
         </ipxact:vendorExtensions>
      </ipxact:file>
   </ipxact:fileSet>
 </ipxact:fileSets>
 <ipxact:description>Increment or decrement in multiples of incr with
variable bit width</ipxact:description>
 <ipxact:parameters>
   <ipxact:parameter kactus2:usageCount="3" maximum="32" minimum="1"
parameterId="uuid_69fca058_a0fb_4e1d_9db2_a713c9781c00" resolve="user"
type="int">
      <ipxact:name>width</ipxact:name>
      <ipxact:value>8</ipxact:value>
   </ipxact:parameter>
   <ipxact:parameter choiceRef="Direction" kactus2:usageCount="1"
parameterId="uuid_74620f8f_d6ae_4da4_b710_5c6c86e460cf" resolve="user"
type="int">
```

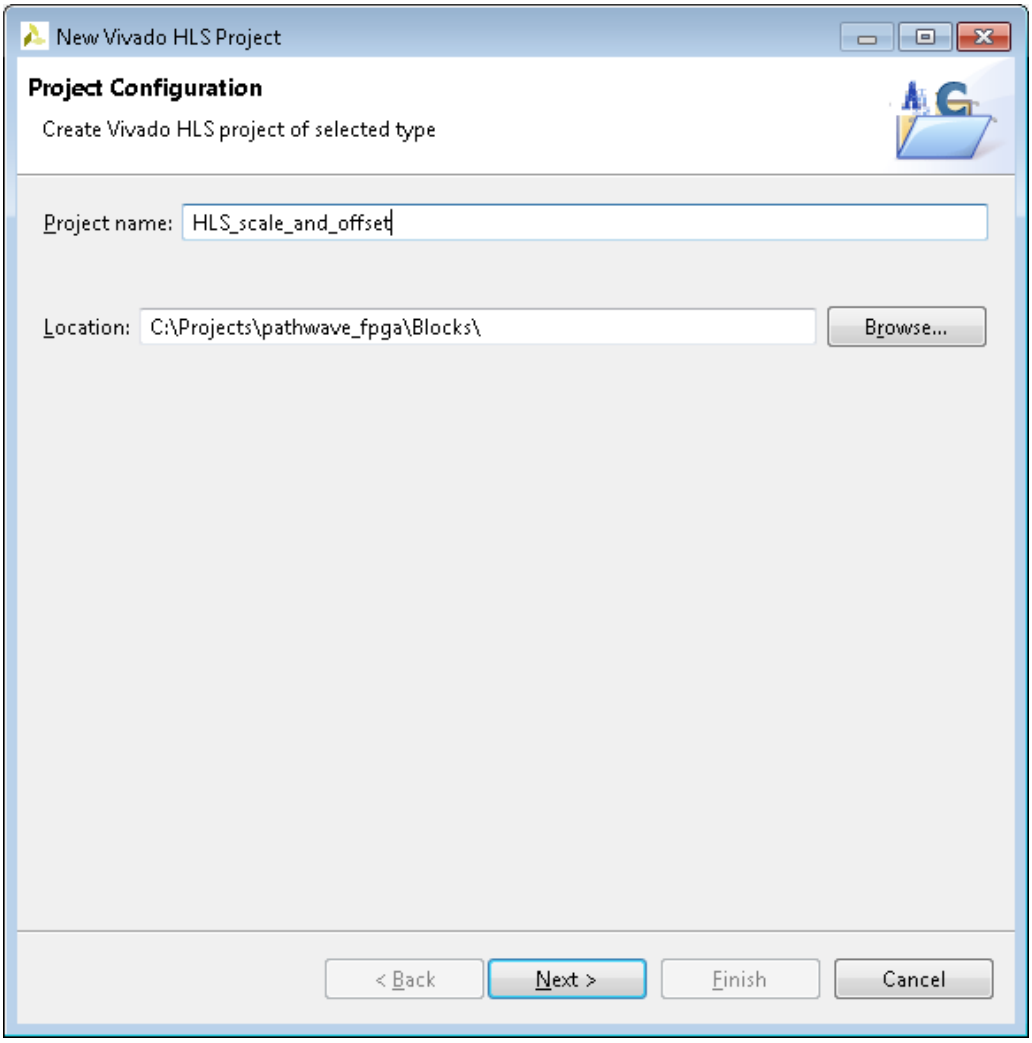
```
        <ipxact:name>dir</ipxact:name>
        <ipxact:displayName>Direction</ipxact:displayName>
        <ipxact:value>0</ipxact:value>
      </ipxact:parameter>
   </ipxact:parameters>
   <ipxact:vendorExtensions>
      <kactus2:author>Keysight</kactus2:author>
      <kactus2:version>3,5,0,0</kactus2:version>
      <kactus2:kts_attributes>
        <kactus2:kts_productHier>Flat</kactus2:kts_productHier>
        <kactus2:kts_implementation>HW</kactus2:kts_implementation>
        <kactus2:kts_firmness>Mutable</kactus2:kts_firmness>
      </kactus2:kts_attributes>
   </ipxact:vendorExtensions>
 </ipxact:component>
```

Import Vivado High-Level Synthesis (HLS) generated HDL with parameterized bus widths using IP-XACT

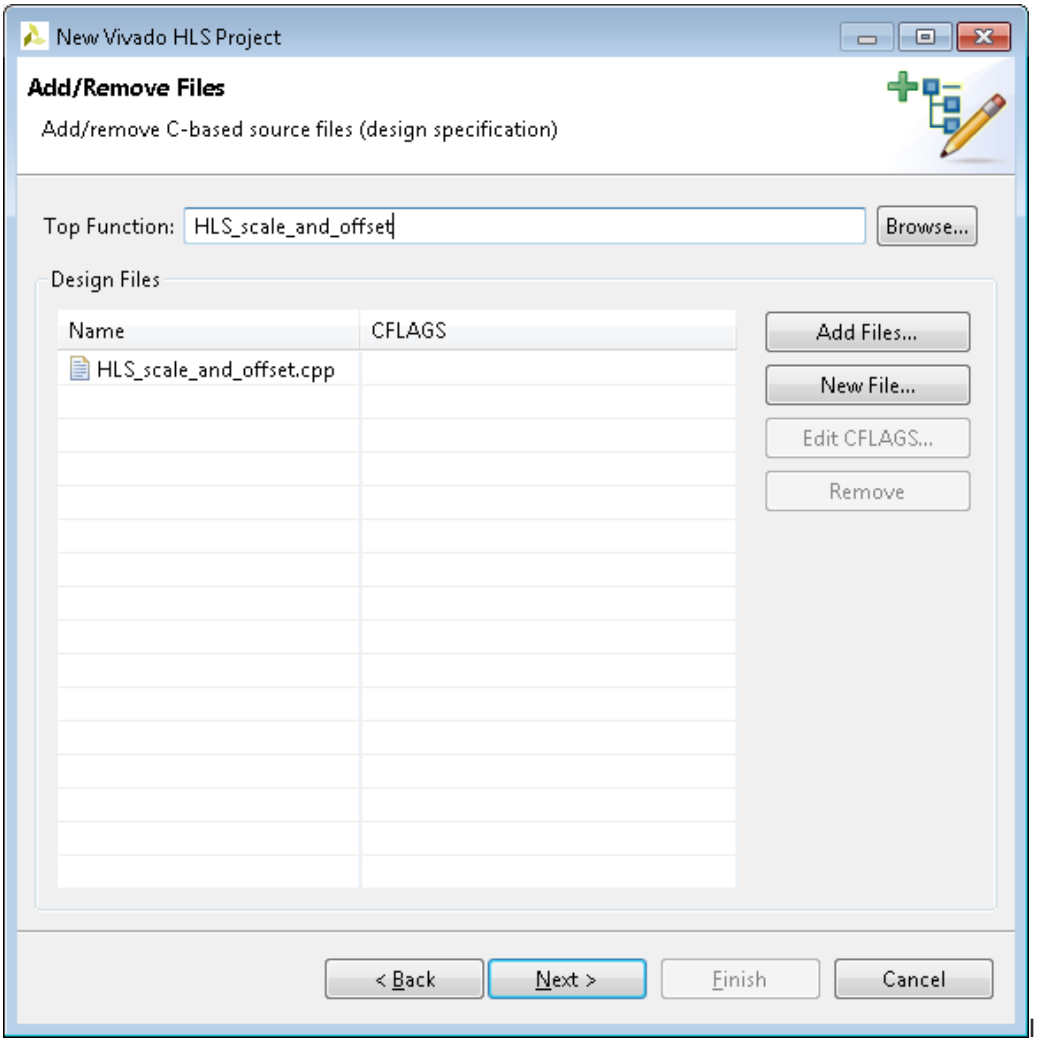
Vivado High-Level Synthesis (HLS) accelerates IP creation by enabling C, C++ and System C specifications to be directly targeted into Xilinx FPGAs without the need to manually create HDL. This tutorial describes the creation of a VHDL design using HLS. The design is a scale and offset circuit. The input and output data streams use an AXIS interface. The scale and offset are programmable via an AXILite interface.

To create the HLS project, first start Vivado High-Level Synthesis (Start → All Programs → Xilinx Design Tools → Vivado 2017.3 → Vivado HLS 2017.3).

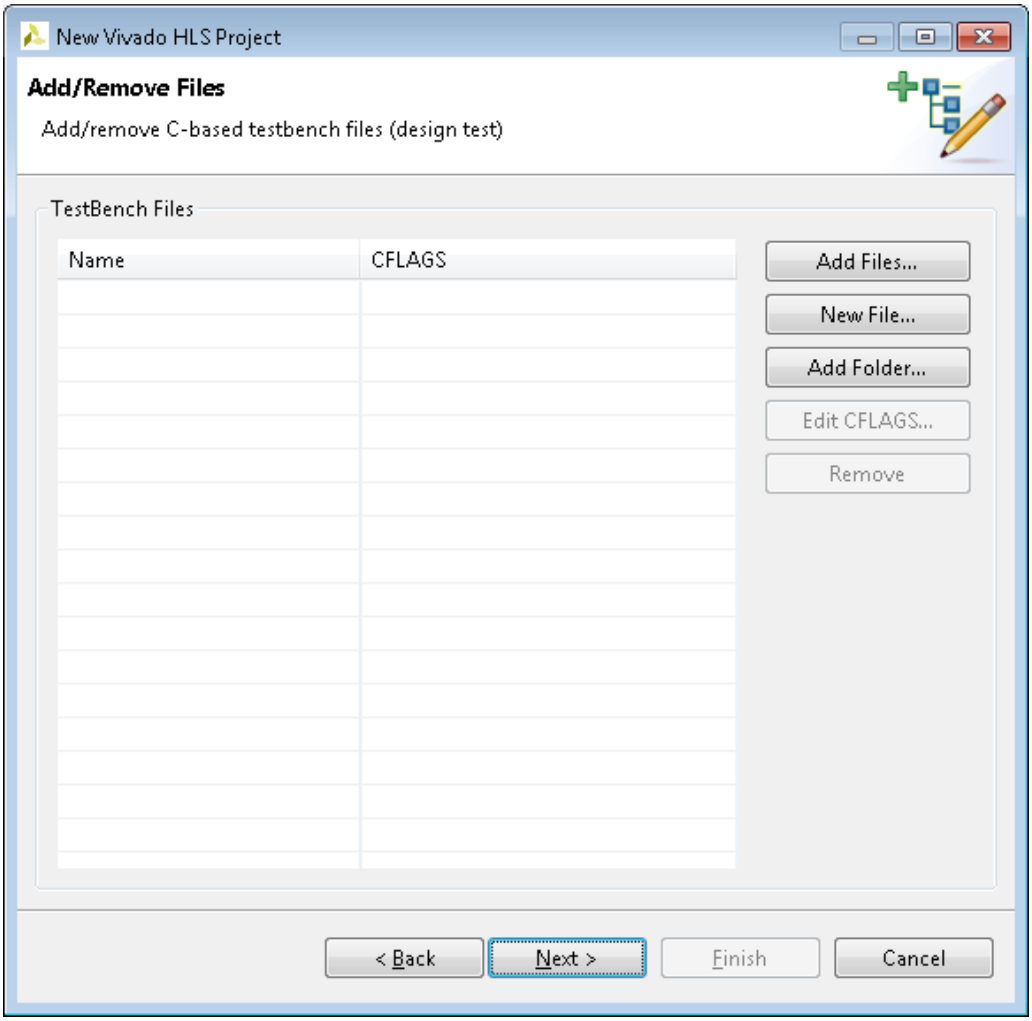
Click on Create New Project, then set the project name to HLS_scale_and_offset as shown in the figure below. Then click Next.



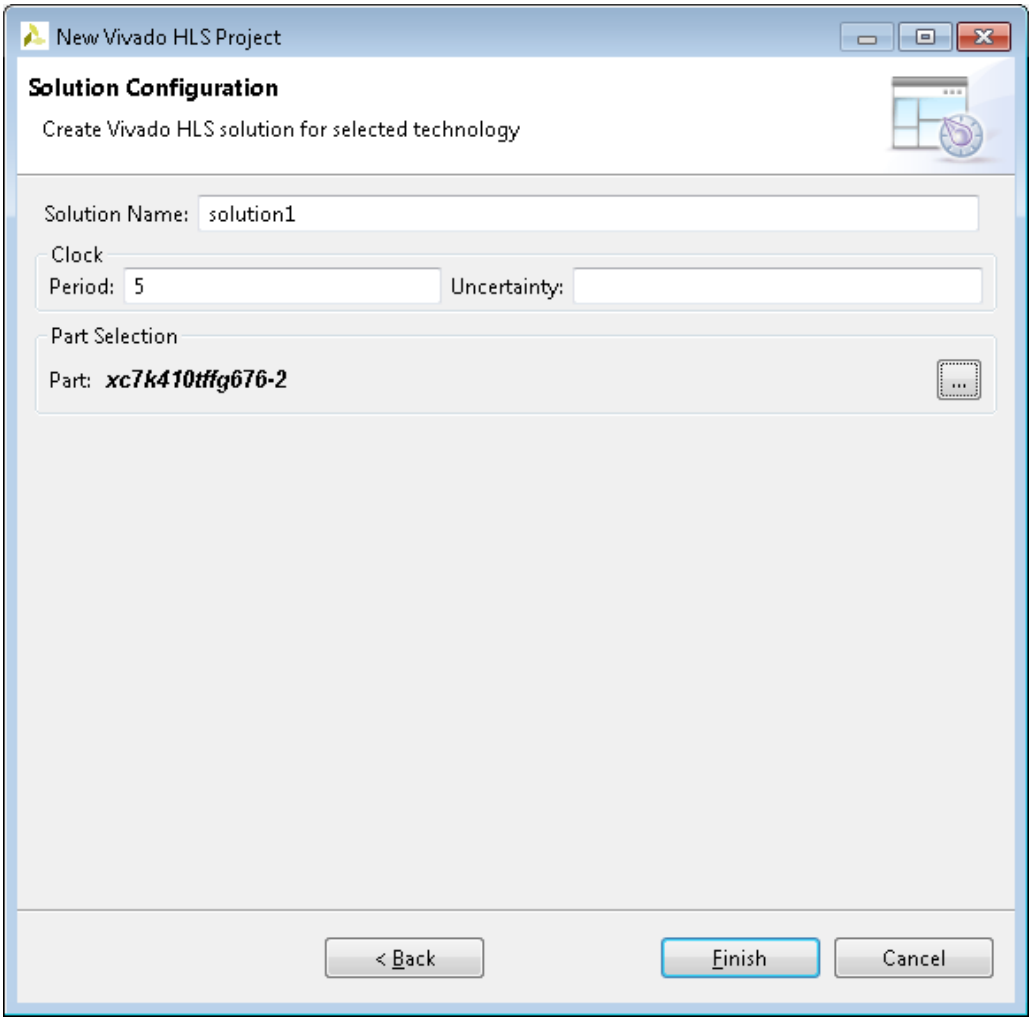
In the next window click on New File, name the file HLS_scale_and_offset.cpp and save it in the same location as the HLS project. Then set the top function to HLS_scale_and_offset as shown in the figure below. Then click Next.



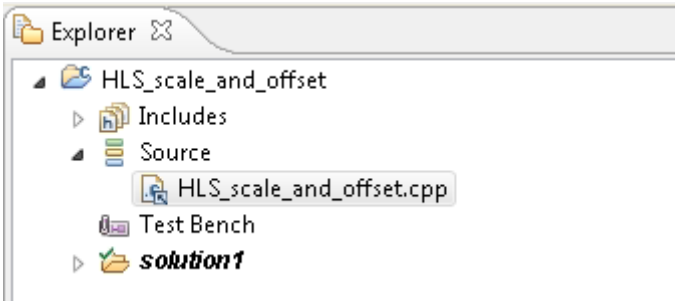
In the next window click Next.



In the next window set the clock period to 5 (200 MHz clock) and set the part to xc7k410tffg676-2 for the M3202A as shown in the figure below. Then click Finish.



In the Explorer window double click on HLS_scale_and_offset.cpp.



Use the code shown in the figure below for HLS_scale_and_offset.cpp:

Code Block 10 HSL_scale_and_offset.cpp

```
#include <ap_fixed.h>
#include <hls_stream.h>


using namespace hls;

typedef ap_fixed<16, 1, AP_TRN, AP_SAT> SAMPLE_T;
typedef stream<SAMPLE_T> SAMPLE_FIFO_T;

void HLS_scale_and_offset(SAMPLE_FIFO_T data,
                          SAMPLE_T scale,
```

```
                            SAMPLE_T offset,
                            SAMPLE_FIFO_T output)
{
#pragma HLS PIPELINE II=1 enable_flush
#pragma HLS INTERFACE axis register both port=output
#pragma HLS INTERFACE axis register both port=data
#pragma HLS INTERFACE s_axilite register port=scale
#pragma HLS INTERFACE s_axilite register port=offset
#pragma HLS INTERFACE s_axilite port=return

    SAMPLE_T product;

    data >> product;
    product = (product * scale + offset);
    output << product;
}
```

Next click the C Synthesis button. HLS generates three VHDL files in the solution1/syn/vhdl folder:

1. HLS_scale_and_offbkb.vhd
2. HLS_scale_and_offset.vhd
3. HLS_scale_and_offset_AXILiteS_s_axi.vhd

HLS_scale_and_offset.vhd is the top level VHDL file. The top level VHDL file generated by HLS is not compatible with Kactus2.

Change HLS_scale_and_offset.vhd line 43 from this:

```
end;
```

To this in order to make it compatible with Kactus2:

```
end HLS_scale_and_offset;
```

Also move HLS_scale_and_offset.vhd line 19 to line 23.

The HLS_scale_and_offset.vhd entity declaration should now match the figure below.

```
12  entity HLS_scale_and_offset is
13  generic (
14      C_S_AXI_AXILITES_ADDR_WIDTH : INTEGER := 5;
15      C_S_AXI_AXILITES_DATA_WIDTH : INTEGER := 32 );
16  port (
17      ap_clk : IN STD_LOGIC;
18      ap_rst_n : IN STD_LOGIC;
19      data_V_V_TDATA : IN STD_LOGIC_VECTOR (15 downto 0);
20      data_V_V_TVALID : IN STD_LOGIC;
21      data_V_V_TREADY : OUT STD_LOGIC;
22      output_V_V_TREADY : IN STD_LOGIC;
23      output_V_V_TDATA : OUT STD_LOGIC_VECTOR (15 downto 0);
24      output_V_V_TVALID : OUT STD_LOGIC;
25      s_axi_AXILiteS_AWVALID : IN STD_LOGIC;
26      s_axi_AXILiteS_AWREADY : OUT STD_LOGIC;
27      s_axi_AXILiteS_AWADDR : IN STD_LOGIC_VECTOR (C_S_AXI_AXILITES_ADDR_WIDTH-1 downto 0);
28      s_axi_AXILiteS_WVALID : IN STD_LOGIC;
29      s_axi_AXILiteS_WREADY : OUT STD_LOGIC;
30      s_axi_AXILiteS_WDATA : IN STD_LOGIC_VECTOR (C_S_AXI_AXILITES_DATA_WIDTH-1 downto 0);
31      s_axi_AXILiteS_WSTRB : IN STD_LOGIC_VECTOR (C_S_AXI_AXILITES_DATA_WIDTH/8-1 downto 0);
32      s_axi_AXILiteS_ARVALID : IN STD_LOGIC;
33      s_axi_AXILiteS_ARREADY : OUT STD_LOGIC;
34      s_axi_AXILiteS_ARADDR : IN STD_LOGIC_VECTOR (C_S_AXI_AXILITES_ADDR_WIDTH-1 downto 0);
35      s_axi_AXILiteS_RVALID : OUT STD_LOGIC;
36      s_axi_AXILiteS_RREADY : IN STD_LOGIC;
37      s_axi_AXILiteS_RDATA : OUT STD_LOGIC_VECTOR (C_S_AXI_AXILITES_DATA_WIDTH-1 downto 0);
38      s_axi_AXILiteS_RRESP : OUT STD_LOGIC_VECTOR (1 downto 0);
39      s_axi_AXILiteS_BVALID : OUT STD_LOGIC;
40      s_axi_AXILiteS_BREADY : IN STD_LOGIC;
41      s_axi_AXILiteS_BRESP : OUT STD_LOGIC_VECTOR (1 downto 0);
42      interrupt : OUT STD_LOGIC );
43  end HLS_scale_and_offset;
```

The following ports are the input AXIS interface:

data_V_V_TDATA : IN STD_LOGIC_VECTOR (15 downto 0);
data_V_V_TVALID : IN STD_LOGIC;
data_V_V_TREADY : OUT STD_LOGIC;

The following ports are the output AXIS interface:

output_V_V_TDATA : OUT STD_LOGIC_VECTOR (15 downto 0);
output_V_V_TVALID : OUT STD_LOGIC;
output_V_V_TREADY : IN STD_LOGIC;

The following ports are the AXILite interface:

s_axi_AXILiteS_AWVALID : IN STD_LOGIC;
s_axi_AXILiteS_AWREADY : OUT STD_LOGIC;
s_axi_AXILiteS_AWADDR : IN STD_LOGIC_VECTOR (C_S_AXI_AXILITES_ADDR_WIDTH-1 downto 0);
s_axi_AXILiteS_WVALID : IN STD_LOGIC;
s_axi_AXILiteS_WREADY : OUT STD_LOGIC;
s_axi_AXILiteS_WDATA : IN STD_LOGIC_VECTOR (C_S_AXI_AXILITES_DATA_WIDTH-1 downto 0);
s_axi_AXILiteS_WSTRB : IN STD_LOGIC_VECTOR (C_S_AXI_AXILITES_DATA_WIDTH/8-1 downto 0);
s_axi_AXILiteS_ARVALID : IN STD_LOGIC;
s_axi_AXILiteS_ARREADY : OUT STD_LOGIC;
s_axi_AXILiteS_ARADDR : IN STD_LOGIC_VECTOR (C_S_AXI_AXILITES_ADDR_WIDTH-1 downto 0);
s_axi_AXILiteS_RVALID : OUT STD_LOGIC;
s_axi_AXILiteS_RREADY : IN STD_LOGIC;
s_axi_AXILiteS_RDATA : OUT STD_LOGIC_VECTOR (C_S_AXI_AXILITES_DATA_WIDTH-1 downto 0);
s_axi_AXILiteS_RRESP : OUT STD_LOGIC_VECTOR (1 downto 0);
s_axi_AXILiteS_BVALID : OUT STD_LOGIC;

s_axi_AXILiteS_BREADY : IN STD_LOGIC;
s_axi_AXILiteS_BRESP : OUT STD_LOGIC_VECTOR (1 downto 0);

Follow the instructions in the Import HDL with parameterized bus widths using IP-XACT tutorial in order to use Kactus2 to generate an IP-XACT file compatible with PathWave FPGA.

The generated IP-XACT is

Code Block 11 HSL_scale_and_offset.1.0.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ipxact:component xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ipxact="http://www.accellera.org/XMLSchema/IPXACT/1685-2014"
xmlns:kactus2="http://kactus2.cs.tut.fi"
xsi:schemaLocation="http://www.accellera.org/XMLSchema/IPXACT/1685-2014
http://www.accellera.org/XMLSchema/IPXACT/1685-2014/index.xsd">
 <ipxact:vendor>keysight.com</ipxact:vendor>
 <ipxact:library>flat</ipxact:library>
 <ipxact:name>HLS_scale_and_offset</ipxact:name>
 <ipxact:version>1.0</ipxact:version>
 <ipxact:busInterfaces>
    <ipxact:busInterface>
      <ipxact:name>clock</ipxact:name>
      <ipxact:busType vendor="keysight.com" library="interfaces"
name="clock" version="1.0"/>
      <ipxact:abstractionTypes>
         <ipxact:abstractionType>
            <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="clock.absDef" version="1.0"/>
            <ipxact:portMaps>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>clk</ipxact:name>
                     <ipxact:range>
                        <ipxact:left>0</ipxact:left>
                        <ipxact:right>0</ipxact:right>
                     </ipxact:range>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>ap_clk</ipxact:name>
                     <ipxact:partSelect>
                        <ipxact:range>
                           <ipxact:left>0</ipxact:left>
                           <ipxact:right>0</ipxact:right>
                        </ipxact:range>
                     </ipxact:partSelect>
                  </ipxact:physicalPort>
               </ipxact:portMap>
            </ipxact:portMaps>
         </ipxact:abstractionType>
      </ipxact:abstractionTypes>
      <ipxact:slave/>
    </ipxact:busInterface>
    <ipxact:busInterface>
      <ipxact:name>nRst</ipxact:name>
      <ipxact:busType vendor="keysight.com" library="interfaces"
name="nRst" version="1.0"/>
      <ipxact:abstractionTypes>
         <ipxact:abstractionType>
            <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="nRst.absDef" version="1.0"/>
            <ipxact:portMaps>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>nRst</ipxact:name>
```

```
                  <ipxact:range>
                     <ipxact:left>0</ipxact:left>
                     <ipxact:right>0</ipxact:right>
                  </ipxact:range>
               </ipxact:logicalPort>
               <ipxact:physicalPort>
                  <ipxact:name>ap_rst_n</ipxact:name>
                  <ipxact:partSelect>
                     <ipxact:range>
                        <ipxact:left>0</ipxact:left>
                        <ipxact:right>0</ipxact:right>
                     </ipxact:range>
                  </ipxact:partSelect>
               </ipxact:physicalPort>
            </ipxact:portMap>
         </ipxact:portMaps>
      </ipxact:abstractionType>
   </ipxact:abstractionTypes>
   <ipxact:slave/>
</ipxact:busInterface>
<ipxact:busInterface>
   <ipxact:name>data_in</ipxact:name>
   <ipxact:busType vendor="keysight.com" library="interfaces"
name="axis" version="1.0"/>
   <ipxact:abstractionTypes>
      <ipxact:abstractionType>
         <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="axis.absDef" version="1.0"/>
         <ipxact:portMaps>
            <ipxact:portMap>
               <ipxact:logicalPort>
                  <ipxact:name>tdata</ipxact:name>
               </ipxact:logicalPort>
               <ipxact:physicalPort>
                  <ipxact:name>data_V_V_TDATA</ipxact:name>
               </ipxact:physicalPort>
            </ipxact:portMap>
            <ipxact:portMap>
               <ipxact:logicalPort>
                  <ipxact:name>tvalid</ipxact:name>
               </ipxact:logicalPort>
               <ipxact:physicalPort>
                  <ipxact:name>data_V_V_TVALID</ipxact:name>
               </ipxact:physicalPort>
            </ipxact:portMap>
            <ipxact:portMap>
               <ipxact:logicalPort>
                  <ipxact:name>tready</ipxact:name>
               </ipxact:logicalPort>
               <ipxact:physicalPort>
                  <ipxact:name>data_V_V_TREADY</ipxact:name>
               </ipxact:physicalPort>
            </ipxact:portMap>
         </ipxact:portMaps>
      </ipxact:abstractionType>
   </ipxact:abstractionTypes>
   <ipxact:slave/>
</ipxact:busInterface>
<ipxact:busInterface>
   <ipxact:name>data_out</ipxact:name>
   <ipxact:busType vendor="keysight.com" library="interfaces"
name="axis" version="1.0"/>
   <ipxact:abstractionTypes>
      <ipxact:abstractionType>
         <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="axis.absDef" version="1.0"/>
```

```
            <ipxact:portMaps>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>tdata</ipxact:name>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>output_V_V_TDATA</ipxact:name>
                  </ipxact:physicalPort>
               </ipxact:portMap>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>tvalid</ipxact:name>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>output_V_V_TVALID</ipxact:name>
                  </ipxact:physicalPort>
               </ipxact:portMap>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>tready</ipxact:name>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>output_V_V_TREADY</ipxact:name>
                  </ipxact:physicalPort>
               </ipxact:portMap>
            </ipxact:portMaps>
         </ipxact:abstractionType>
      </ipxact:abstractionTypes>
      <ipxact:master/>
   </ipxact:busInterface>
   <ipxact:busInterface>
      <ipxact:name>interrupt</ipxact:name>
      <ipxact:busType vendor="keysight.com" library="interfaces"
name="wire" version="1.0"/>
      <ipxact:abstractionTypes>
         <ipxact:abstractionType>
            <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="wire.absDef" version="1.0"/>
            <ipxact:portMaps>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>wire</ipxact:name>
                     <ipxact:range>
                        <ipxact:left>0</ipxact:left>
                        <ipxact:right>0</ipxact:right>
                     </ipxact:range>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>interrupt</ipxact:name>
                     <ipxact:partSelect>
                        <ipxact:range>
                           <ipxact:left>0</ipxact:left>
                           <ipxact:right>0</ipxact:right>
                        </ipxact:range>
                     </ipxact:partSelect>
                  </ipxact:physicalPort>
               </ipxact:portMap>
            </ipxact:portMaps>
         </ipxact:abstractionType>
      </ipxact:abstractionTypes>
      <ipxact:master/>
   </ipxact:busInterface>
   <ipxact:busInterface>
      <ipxact:name>axilite</ipxact:name>
      <ipxact:busType vendor="keysight.com" library="interfaces"
name="axilite" version="1.0"/>
```

```
      <ipxact:abstractionTypes>
         <ipxact:abstractionType>
            <ipxact:abstractionRef vendor="keysight.com"
library="interfaces" name="axilite.absDef" version="1.0"/>
            <ipxact:portMaps>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>awready</ipxact:name>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>s_axi_AXILiteS_AWREADY</ipxact:name>
                  </ipxact:physicalPort>
               </ipxact:portMap>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>awaddr</ipxact:name>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>s_axi_AXILiteS_AWADDR</ipxact:name>
                  </ipxact:physicalPort>
               </ipxact:portMap>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>wvalid</ipxact:name>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>s_axi_AXILiteS_WVALID</ipxact:name>
                  </ipxact:physicalPort>
               </ipxact:portMap>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>awvalid</ipxact:name>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>s_axi_AXILiteS_AWVALID</ipxact:name>
                  </ipxact:physicalPort>
               </ipxact:portMap>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>wready</ipxact:name>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>s_axi_AXILiteS_WREADY</ipxact:name>
                  </ipxact:physicalPort>
               </ipxact:portMap>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>wdata</ipxact:name>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>s_axi_AXILiteS_WDATA</ipxact:name>
                  </ipxact:physicalPort>
               </ipxact:portMap>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>wstrb</ipxact:name>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
                     <ipxact:name>s_axi_AXILiteS_WSTRB</ipxact:name>
                  </ipxact:physicalPort>
               </ipxact:portMap>
               <ipxact:portMap>
                  <ipxact:logicalPort>
                     <ipxact:name>arvalid</ipxact:name>
                  </ipxact:logicalPort>
                  <ipxact:physicalPort>
```

```
            <ipxact:name>s_axi_AXILiteS_ARVALID</ipxact:name>
         </ipxact:physicalPort>
      </ipxact:portMap>
      <ipxact:portMap>
         <ipxact:logicalPort>
            <ipxact:name>arready</ipxact:name>
         </ipxact:logicalPort>
         <ipxact:physicalPort>
            <ipxact:name>s_axi_AXILiteS_ARREADY</ipxact:name>
         </ipxact:physicalPort>
      </ipxact:portMap>
      <ipxact:portMap>
         <ipxact:logicalPort>
            <ipxact:name>araddr</ipxact:name>
         </ipxact:logicalPort>
         <ipxact:physicalPort>
            <ipxact:name>s_axi_AXILiteS_ARADDR</ipxact:name>
         </ipxact:physicalPort>
      </ipxact:portMap>
      <ipxact:portMap>
         <ipxact:logicalPort>
            <ipxact:name>rvalid</ipxact:name>
         </ipxact:logicalPort>
         <ipxact:physicalPort>
            <ipxact:name>s_axi_AXILiteS_RVALID</ipxact:name>
         </ipxact:physicalPort>
      </ipxact:portMap>
      <ipxact:portMap>
         <ipxact:logicalPort>
            <ipxact:name>rready</ipxact:name>
         </ipxact:logicalPort>
         <ipxact:physicalPort>
            <ipxact:name>s_axi_AXILiteS_RREADY</ipxact:name>
         </ipxact:physicalPort>
      </ipxact:portMap>
      <ipxact:portMap>
         <ipxact:logicalPort>
            <ipxact:name>rdata</ipxact:name>
         </ipxact:logicalPort>
         <ipxact:physicalPort>
            <ipxact:name>s_axi_AXILiteS_RDATA</ipxact:name>
         </ipxact:physicalPort>
      </ipxact:portMap>
      <ipxact:portMap>
         <ipxact:logicalPort>
            <ipxact:name>rresp</ipxact:name>
         </ipxact:logicalPort>
         <ipxact:physicalPort>
            <ipxact:name>s_axi_AXILiteS_RRESP</ipxact:name>
         </ipxact:physicalPort>
      </ipxact:portMap>
      <ipxact:portMap>
         <ipxact:logicalPort>
            <ipxact:name>bvalid</ipxact:name>
         </ipxact:logicalPort>
         <ipxact:physicalPort>
            <ipxact:name>s_axi_AXILiteS_BVALID</ipxact:name>
         </ipxact:physicalPort>
      </ipxact:portMap>
      <ipxact:portMap>
         <ipxact:logicalPort>
            <ipxact:name>bready</ipxact:name>
         </ipxact:logicalPort>
         <ipxact:physicalPort>
            <ipxact:name>s_axi_AXILiteS_BREADY</ipxact:name>
         </ipxact:physicalPort>
```

```
              </ipxact:portMap>
              <ipxact:portMap>
                 <ipxact:logicalPort>
                    <ipxact:name>bresp</ipxact:name>
                 </ipxact:logicalPort>
                 <ipxact:physicalPort>
                    <ipxact:name>s_axi_AXILiteS_BRESP</ipxact:name>
                 </ipxact:physicalPort>
              </ipxact:portMap>
           </ipxact:portMaps>
        </ipxact:abstractionType>
     </ipxact:abstractionTypes>
     <ipxact:slave/>
   </ipxact:busInterface>
 </ipxact:busInterfaces>
 <ipxact:model>
   <ipxact:views>
     <ipxact:view>
        <ipxact:name>flat_vhdl</ipxact:name>
        <ipxact:envIdentifier>VHDL:Kactus2:</ipxact:envIdentifier>

 <ipxact:componentInstantiationRef>vhdl_implementation</ipxact:componentI
nstantiationRef>
     </ipxact:view>
   </ipxact:views>
   <ipxact:instantiations>
     <ipxact:componentInstantiation>
        <ipxact:name>vhdl_implementation</ipxact:name>
        <ipxact:language>VHDL</ipxact:language>
        <ipxact:moduleName>HLS_scale_and_offset</ipxact:moduleName>
        <ipxact:architectureName>behav</ipxact:architectureName>
        <ipxact:moduleParameters>
           <ipxact:moduleParameter dataType="INTEGER"
parameterId="uuid_1f076b4a_4ddc_4d5c_b810_1bfb6813560d"
usageType="nontyped">
              <ipxact:name>C_S_AXI_AXILITES_ADDR_WIDTH</ipxact:name>

 <ipxact:value>uuid_feaace4f_64fe_4b3e_b2fb_2c1c86f7118b</ipxact:value>
           </ipxact:moduleParameter>
           <ipxact:moduleParameter dataType="INTEGER"
parameterId="uuid_7068d89f_b6ef_4747_8c06_b97a052832c2"
usageType="nontyped">
              <ipxact:name>C_S_AXI_AXILITES_DATA_WIDTH</ipxact:name>

 <ipxact:value>uuid_6c5c1791_aa38_4f75_b60b_e734ab4bf622</ipxact:value>
           </ipxact:moduleParameter>
        </ipxact:moduleParameters>
        <ipxact:fileSetRef>
           <ipxact:localName>synthesis</ipxact:localName>
        </ipxact:fileSetRef>
     </ipxact:componentInstantiation>
   </ipxact:instantiations>
   <ipxact:ports>
     <ipxact:port>
        <ipxact:name>ap_clk</ipxact:name>
        <ipxact:wire>
           <ipxact:direction>in</ipxact:direction>
           <ipxact:wireTypeDefs>
              <ipxact:wireTypeDef>
                 <ipxact:typeName>STD_LOGIC</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
              </ipxact:wireTypeDef>
           </ipxact:wireTypeDefs>
        </ipxact:wire>
     </ipxact:port>
```

```
      <ipxact:port>
         <ipxact:name>ap_rst_n</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>

<ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>data_V_V_TDATA</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:vectors>
               <ipxact:vector>
                  <ipxact:left>15</ipxact:left>
                  <ipxact:right>0</ipxact:right>
               </ipxact:vector>
            </ipxact:vectors>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC_VECTOR</ipxact:typeName>

<ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>data_V_V_TVALID</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>

<ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>data_V_V_TREADY</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>out</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>

<ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>output_V_V_TREADY</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>
```

```
<ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
             </ipxact:wireTypeDef>
          </ipxact:wireTypeDefs>
       </ipxact:wire>
    </ipxact:port>
    <ipxact:port>
       <ipxact:name>output_V_V_TDATA</ipxact:name>
       <ipxact:wire>
          <ipxact:direction>out</ipxact:direction>
          <ipxact:vectors>
             <ipxact:vector>
                <ipxact:left>15</ipxact:left>
                <ipxact:right>0</ipxact:right>
             </ipxact:vector>
          </ipxact:vectors>
          <ipxact:wireTypeDefs>
             <ipxact:wireTypeDef>
                <ipxact:typeName>STD_LOGIC_VECTOR</ipxact:typeName>

<ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
             </ipxact:wireTypeDef>
          </ipxact:wireTypeDefs>
       </ipxact:wire>
    </ipxact:port>
    <ipxact:port>
       <ipxact:name>output_V_V_TVALID</ipxact:name>
       <ipxact:wire>
          <ipxact:direction>out</ipxact:direction>
          <ipxact:wireTypeDefs>
             <ipxact:wireTypeDef>
                <ipxact:typeName>STD_LOGIC</ipxact:typeName>

<ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
             </ipxact:wireTypeDef>
          </ipxact:wireTypeDefs>
       </ipxact:wire>
    </ipxact:port>
    <ipxact:port>
       <ipxact:name>s_axi_AXILiteS_AWVALID</ipxact:name>
       <ipxact:wire>
          <ipxact:direction>in</ipxact:direction>
          <ipxact:wireTypeDefs>
             <ipxact:wireTypeDef>
                <ipxact:typeName>STD_LOGIC</ipxact:typeName>

<ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
             </ipxact:wireTypeDef>
          </ipxact:wireTypeDefs>
       </ipxact:wire>
    </ipxact:port>
    <ipxact:port>
       <ipxact:name>s_axi_AXILiteS_AWREADY</ipxact:name>
       <ipxact:wire>
          <ipxact:direction>out</ipxact:direction>
          <ipxact:wireTypeDefs>
             <ipxact:wireTypeDef>
                <ipxact:typeName>STD_LOGIC</ipxact:typeName>

<ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
             </ipxact:wireTypeDef>
          </ipxact:wireTypeDefs>
       </ipxact:wire>
    </ipxact:port>
    <ipxact:port>
       <ipxact:name>s_axi_AXILiteS_AWADDR</ipxact:name>
       <ipxact:wire>
```

```
            <ipxact:direction>in</ipxact:direction>
            <ipxact:vectors>
               <ipxact:vector>
                  <ipxact:left>uuid_feaace4f_64fe_4b3e_b2fb_2c1c86f7118b-
1</ipxact:left>
                  <ipxact:right>0</ipxact:right>
               </ipxact:vector>
            </ipxact:vectors>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC_VECTOR</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_WVALID</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_WREADY</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>out</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_WDATA</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:vectors>
               <ipxact:vector>
                  <ipxact:left>uuid_6c5c1791_aa38_4f75_b60b_e734ab4bf622-
1</ipxact:left>
                  <ipxact:right>0</ipxact:right>
               </ipxact:vector>
            </ipxact:vectors>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC_VECTOR</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_WSTRB</ipxact:name>
         <ipxact:wire>
```

```
            <ipxact:direction>in</ipxact:direction>
            <ipxact:vectors>
               <ipxact:vector>

 <ipxact:left>uuid_6c5c1791_aa38_4f75_b60b_e734ab4bf622/8-1</ipxact:left>
                  <ipxact:right>0</ipxact:right>
               </ipxact:vector>
            </ipxact:vectors>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC_VECTOR</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_ARVALID</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_ARREADY</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>out</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_ARADDR</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:vectors>
               <ipxact:vector>
                  <ipxact:left>uuid_feaace4f_64fe_4b3e_b2fb_2c1c86f7118b-
1</ipxact:left>
                  <ipxact:right>0</ipxact:right>
               </ipxact:vector>
            </ipxact:vectors>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC_VECTOR</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_RVALID</ipxact:name>
         <ipxact:wire>
```

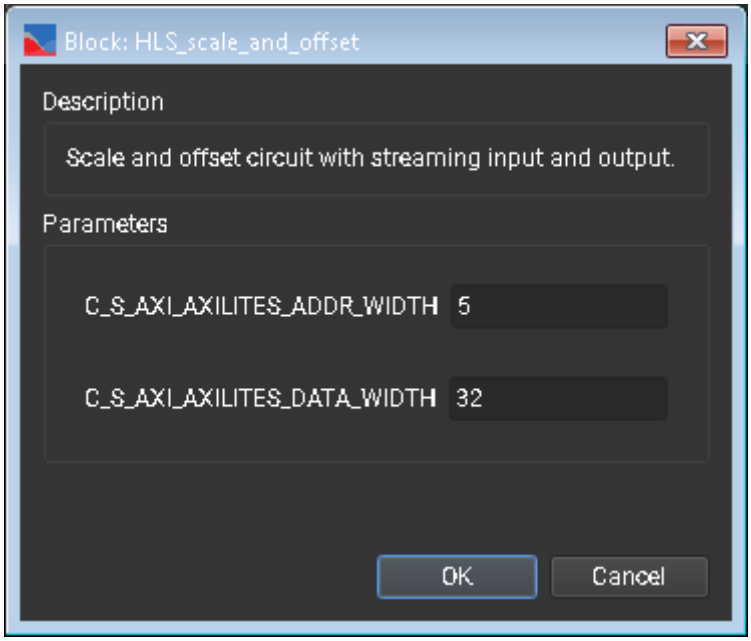
```
            <ipxact:direction>out</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_RREADY</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_RDATA</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>out</ipxact:direction>
            <ipxact:vectors>
               <ipxact:vector>
                  <ipxact:left>uuid_6c5c1791_aa38_4f75_b60b_e734ab4bf622-
1</ipxact:left>
                  <ipxact:right>0</ipxact:right>
               </ipxact:vector>
            </ipxact:vectors>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC_VECTOR</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_RRESP</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>out</ipxact:direction>
            <ipxact:vectors>
               <ipxact:vector>
                  <ipxact:left>1</ipxact:left>
                  <ipxact:right>0</ipxact:right>
               </ipxact:vector>
            </ipxact:vectors>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC_VECTOR</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_BVALID</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>out</ipxact:direction>
```

```
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_BREADY</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>in</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>s_axi_AXILiteS_BRESP</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>out</ipxact:direction>
            <ipxact:vectors>
               <ipxact:vector>
                  <ipxact:left>1</ipxact:left>
                  <ipxact:right>0</ipxact:right>
               </ipxact:vector>
            </ipxact:vectors>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC_VECTOR</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
      <ipxact:port>
         <ipxact:name>interrupt</ipxact:name>
         <ipxact:wire>
            <ipxact:direction>out</ipxact:direction>
            <ipxact:wireTypeDefs>
               <ipxact:wireTypeDef>
                  <ipxact:typeName>STD_LOGIC</ipxact:typeName>

 <ipxact:typeDefinition>IEEE.std_logic_1164.all</ipxact:typeDefinition>
               </ipxact:wireTypeDef>
            </ipxact:wireTypeDefs>
         </ipxact:wire>
      </ipxact:port>
   </ipxact:ports>
 </ipxact:model>
 <ipxact:fileSets>
   <ipxact:fileSet>
      <ipxact:name>synthesis</ipxact:name>
      <ipxact:file>

 <ipxact:name>../solution1/syn/vhdl/HLS_scale_and_offbkb.vhd</ipxact:name
>
         <ipxact:fileType>vhdlSource</ipxact:fileType>
         <ipxact:vendorExtensions>
```

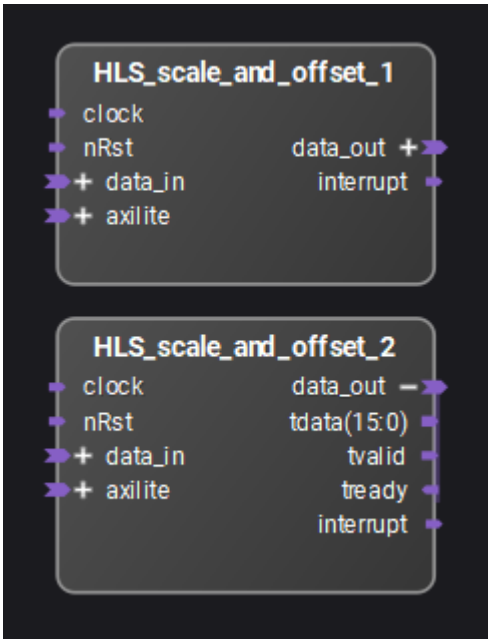
```
 <kactus2:hash>75b7304c4bd4a33acb315af86c07c2a406b2d857</kactus2:hash>
         </ipxact:vendorExtensions>
      </ipxact:file>
      <ipxact:file>

 <ipxact:name>../solution1/syn/vhdl/HLS_scale_and_offset_AXILiteS_s_axi.v
hd</ipxact:name>
         <ipxact:fileType>vhdlSource</ipxact:fileType>
      </ipxact:file>
      <ipxact:file>

 <ipxact:name>../solution1/syn/vhdl/HLS_scale_and_offset.vhd</ipxact:name
>
         <ipxact:fileType>vhdlSource</ipxact:fileType>
      </ipxact:file>
    </ipxact:fileSet>
 </ipxact:fileSets>
 <ipxact:description>Scale and offset circuit with streaming input and
output.</ipxact:description>
 <ipxact:parameters>
    <ipxact:parameter kactus2:usageCount="3"
parameterId="uuid_feaace4f_64fe_4b3e_b2fb_2c1c86f7118b" resolve="user"
type="int">
      <ipxact:name>C_S_AXI_AXILITES_ADDR_WIDTH</ipxact:name>
      <ipxact:value>5</ipxact:value>
    </ipxact:parameter>
    <ipxact:parameter kactus2:usageCount="4"
parameterId="uuid_6c5c1791_aa38_4f75_b60b_e734ab4bf622" resolve="user"
type="int">
      <ipxact:name>C_S_AXI_AXILITES_DATA_WIDTH</ipxact:name>
      <ipxact:value>32</ipxact:value>
    </ipxact:parameter>
 </ipxact:parameters>
 <ipxact:vendorExtensions>
    <kactus2:author>Keysight</kactus2:author>
    <kactus2:sourceDirectories>

 <kactus2:sourceDirectory>../solution1/syn/vhdl</kactus2:sourceDirectory>
    </kactus2:sourceDirectories>
    <kactus2:fileDependencies>
      <kactus2:fileDependency manual="false" bidirectional="false"
locked="false">

 <kactus2:fileRef1>../solution1/syn/vhdl/HLS_scale_and_offbkb.vhd</kactus
2:fileRef1>
         <kactus2:fileRef2>$External$/INTEGER.vhd</kactus2:fileRef2>
         <ipxact:description>Component instantiation for entity
INTEGER</ipxact:description>
      </kactus2:fileDependency>
      <kactus2:fileDependency manual="false" bidirectional="false"
locked="false">

 <kactus2:fileRef1>../solution1/syn/vhdl/HLS_scale_and_offbkb.vhd</kactus
2:fileRef1>
         <kactus2:fileRef2>$External$/IN.vhd</kactus2:fileRef2>
         <ipxact:description>Component instantiation for entity
IN</ipxact:description>
      </kactus2:fileDependency>
      <kactus2:fileDependency manual="false" bidirectional="false"
locked="false">

 <kactus2:fileRef1>../solution1/syn/vhdl/HLS_scale_and_offbkb.vhd</kactus
2:fileRef1>
         <kactus2:fileRef2>$External$/OUT.vhd</kactus2:fileRef2>
         <ipxact:description>Component instantiation for entity
OUT</ipxact:description>
```

```
      </kactus2:fileDependency>
      <kactus2:fileDependency manual="false" bidirectional="false"
locked="false">

 <kactus2:fileRef1>../solution1/syn/vhdl/HLS_scale_and_offbkb.vhd</kactus
2:fileRef1>
         <kactus2:fileRef2>$External$/component.vhd</kactus2:fileRef2>
         <ipxact:description>Component instantiation for entity
component</ipxact:description>
      </kactus2:fileDependency>
    </kactus2:fileDependencies>
    <kactus2:version>3,5,77,0</kactus2:version>
    <kactus2:kts_attributes>
      <kactus2:kts_productHier>Flat</kactus2:kts_productHier>
      <kactus2:kts_implementation>HW</kactus2:kts_implementation>
      <kactus2:kts_firmness>Mutable</kactus2:kts_firmness>
    </kactus2:kts_attributes>
 </ipxact:vendorExtensions>
</ipxact:component>
```

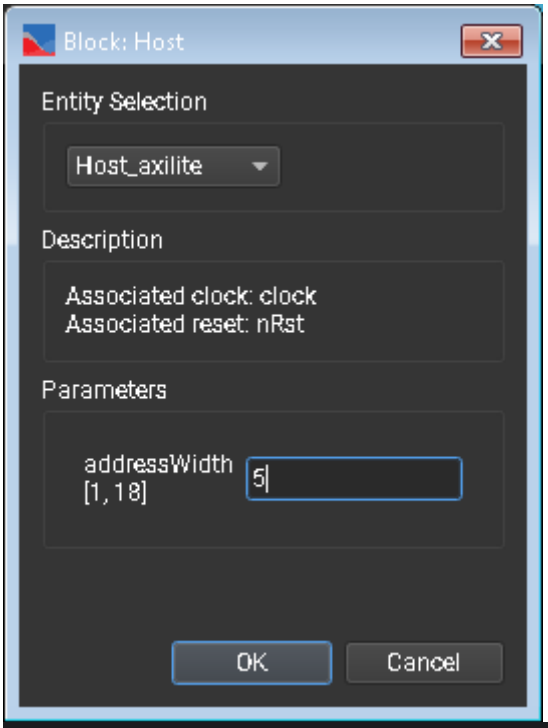
When that block is used within PathWave FPGA, the following dialog box will show up. This shows the description of the IP block along with the user modifiable parameters. In this case there are two parameters, C_S_AXI_AXILITES_ADDR_WIDTH with a default value of 5 and C_S_AXI_AXILITES_DATA_WIDTH with a default value of 32.

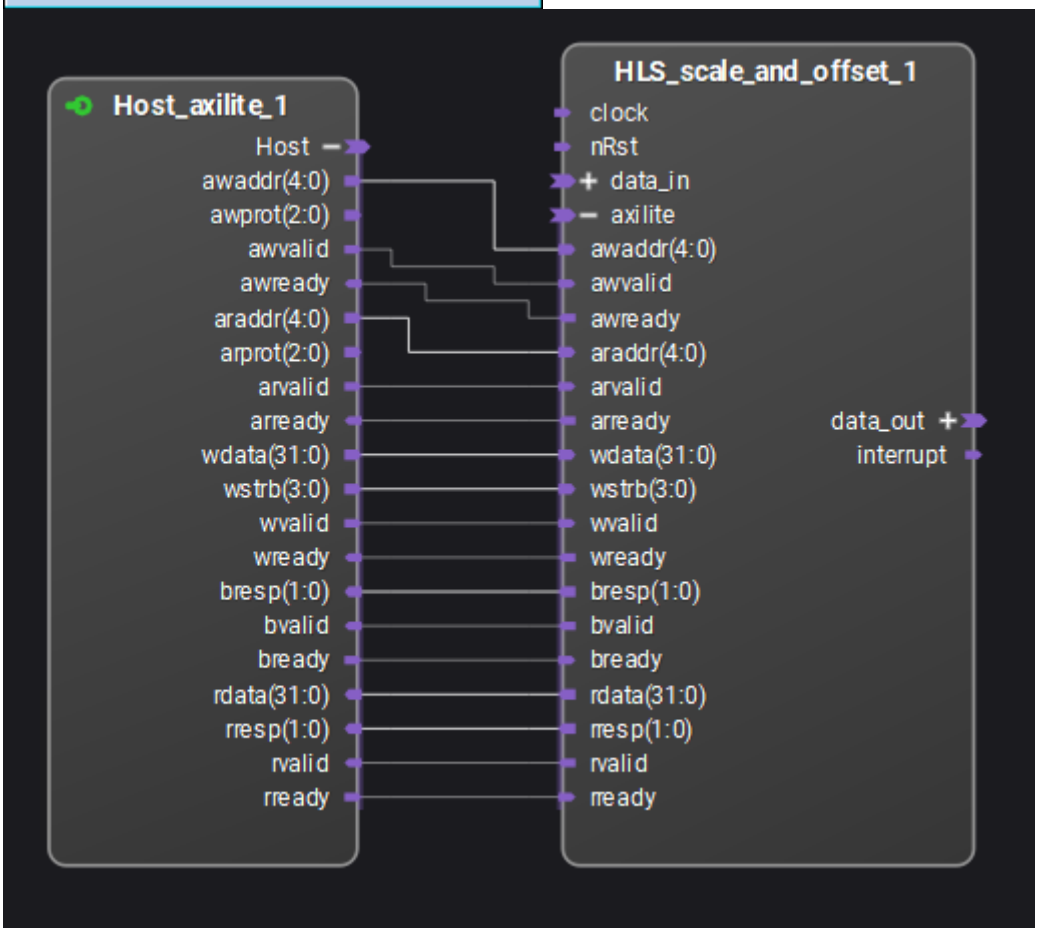


In this screen capture from PathWave FPGA, the instance HLS_scale_and_offset_1 is shown with the *data_out* interface collapsed. The internal ports that make up that interface are not shown and the interface can be connected to other compatible interfaces with one connection. The instance HLS_scale_and_offset_2 is shown with the *data_out* interface expanded to show the internal ports that make up that interface. The entire interface can be connected with one connection by using the *data_out* port or the individual ports within the interface can be connected separately if desired.



The HLS_scale_and_offset axilite interface needs to be connected to a Host MemoryMap block configured for Host_axilite and an address width of 5 as shown in the figure below.


Block: Host
Entity Selection
Host_axilite
Description
Associated clock: clock
Associated reset: nRst
Parameters
addressWidth
[1, 18]
5
OK
Cancel


Host_axilite_1
Host
awaddr(4:0)
awprot(2:0)
awvalid
awready
araddr(4:0)
arprot(2:0)
arvalid
arready
wdata(31:0)
wstrb(3:0)
wvalid
wready
bresp(1:0)
bvalid
bready
rdata(31:0)
rresp(1:0)
rvalid
rready
HLS_scale_and_offset_1
clock
nRst
data_in
axilite
awaddr(4:0)
awvalid
awready
araddr(4:0)
arvalid
arready
wdata(31:0)
wstrb(3:0)
wvalid
wready
bresp(1:0)
bvalid
bready
rdata(31:0)
rresp(1:0)
rvalid
rready
data_out
interrupt

Legal

Portions of this software are licensed by third parties including open source terms and conditions.

7-zip

PathWave FPGA 2018 uses parts of 7-Zip, which is licensed under the GNU LGPL license. The source code may be found at http://www.7-zip.org/

License for use and distribution
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

7-Zip Copyright (C) 1999-2016 Igor Pavlov.

Licenses for files are:

1) 7z.dll: GNU LGPL + unRAR restriction
2) All other files: GNU LGPL

The GNU LGPL + unRAR restriction means that you must follow both GNU LGPL rules and unRAR restriction rules.

Note:
You can use 7-Zip on any computer, including a computer in a commercial organization. You don't need to register or pay for 7-Zip.

GNU LGPL information

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You can receive a copy of the GNU Lesser General Public License from
http://www.gnu.org/

unRAR restriction

The decompression engine for RAR archives was developed using source
code of unRAR program.
All copyrights to original unRAR code are owned by Alexander Roshal.

The license for original unRAR code has the following restriction:

The unRAR sources cannot be used to re-create the RAR compression algorithm,
which is proprietary. Distribution of modified unRAR sources in separate form
or as a part of other software is permitted, provided that it is clearly
stated in the documentation and source comments that the code may
not be used to develop a RAR (WinRAR) compatible archiver.

--
Igor Pavlov

bzip2

PathWave FPGA 2018 uses bzip2 v1.0.6, used with permission. For more information, visit https://spdx.org/licenses/bzip2-1.0.6.html.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Lua

PathWave FPGA 2018 uses parts of Lua 5.3.4.

Qt

PathWave FPGA 2018 uses Qt 5.7.0 and 5.6.2, licensed under the terms of GNU LGPLv3. For more information or to receive a copy of the source code for Qt, visit http://support.keysight.com.

Xerces-C++

PathWave FPGA 2018 uses Xerces-C++ 3.2.0, licensed under the terms of Apache License v2.0, which is displayed below. For more information, visit https://xerces.apache.org/xerces-c/.

zlib

PathWave FPGA 2018 uses zlib 1.2.11, used by permission. For more information, visit https://www.zlib.net/zlib_license.html.

Apache License v2.0

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable

(except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

   You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any

character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: HOW TO APPLY THE APACHE LICENSE TO YOUR WORK

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
See the License for the specific language governing permissions and
limitations under the License.
```

GNU GPLv3

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <https://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU

General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or

without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid

for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then

you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
    <one line to give the program's name and a brief idea of what it
does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or
modify
    it under the terms of the GNU General Public License as published
by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
```

```
    along with this program.  If not, see
<https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper
mail.

If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type
`show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the
appropriate parts of the General Public License. Of course, your
program's commands might be different; for a GUI interface, you would
use an "about box".

You should also get your employer (if you work as a programmer) or
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. For more information on this, and how to apply and follow
the GNU GPL, see <https://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your
program into proprietary programs. If your program is a subroutine
library, you may consider it more useful to permit linking proprietary
applications with the library. If this is what you want to do, use the
GNU Lesser General Public License instead of this License. But first,
please read <https://www.gnu.org/licenses/why-not-lgpl.html>.
```

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <https://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

- a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
- b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

- a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

- a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
- c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
- d) Do one of the following:
    - 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
    - 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

- e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.
- b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.