

Juha Arvio

Kactus 2

MANUAL

Table of Contents

[1. Identification 4](#_Toc307412519)

[2. Version history 5](#_Toc307412520)

[2.1 Documentation 5](#_Toc307412521)

[3. IP-block files 6](#_Toc307412522)

[3.1 Documentation 6](#_Toc307412523)

[3.2 HDL files 6](#_Toc307412524)

[3.3 Software drivers 7](#_Toc307412525)

[3.4 HDL testbench 7](#_Toc307412526)

[3.5 Test systems 7](#_Toc307412527)

[4. Dependencies 8](#_Toc307412528)

[5. General description 9](#_Toc307412529)

[5.1.1 Ports and signals 10](#_Toc307412530)

List of Abbreviations

fifo first in, first out

FPGA Field Programmable Gate Array

Gbps Giga bits per second

H2P HIBI to PC

HDL Hardware Design Language

HIBI Heterogeneous IP Block Interconnect

HW Hardware

ID Identification

I/O Input / Output

IP Intellectual Property

IRQ Interrupt Request

P2H PC to HIBI

PC Personal Computer

SoC System-on-Chip

SW Software

tbd to be decided

TLP Transaction Layer Protocol

TUT Tampere University of Technology

VHDL VHSIC Hardware Description Language

VHSIC Very High Speed Integrated Circuit

# Version history

## Documentation

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Change description | Author |
| 26.10.2011 | 0.1 | Manual documentation started | Juha Arvio |

# Setting up Kactus 2 in Ubuntu Linux

## Installing pre-required software

* Install QT Creator from Ubuntu Software Center
  + Search for QT Creator

## Installing Kactus 2

* Download and extract the source codes (kactus2Linux.[version].tar.gz) from <http://sourceforge.net/projects/kactus2/>
* Open a terminal
  + go to directory kactus2-code/
    - type “qmake Kactus2.pro”
    - type “make”
    - the Kactus 2 executable will be compiled to kactus2-code/executable/

# Download Funbase ip-library

* Download and extract the ip-library package (funbase\_ip\_library\_latest.tar.gz) from http://opencores.org/project,funbase\_ip\_library

# Kactus 2 use example (DE2 demo)

* Launch Kactus 2
  + Open a terminal and go to directory kactus2-code
  + type “LD\_LIBRARY\_PATH=$PWD/GCF/bin executable/Kactus2 &”
* Add the Funbase IP library to Kactus 2
  + Click “Settings” on the toolbar
    - In General set the Default project directory to the trunk directory where you extracted the Funbase IP-library (funbase\_ip\_library/trunk)
    - Confirm the change by pushing “OK”
  + Click the magnifier button in the “Library” group on the toolbar
    - Select the directory TUT and press “Open”
  + Click the magnifier button again
    - Select the directory Flexibilis and press “Open”
* Create a new design
  + Click “New” on the toolbar
    - Select design
      * Your design will be saved under the Default project directory which you earlier set, but you can optionally change this root directory before you type the VLNV



* + - * Set product hierarchy to “SoC”
      * Set firmness to “Fixed”
      * Fill in Vendor, Name and Version (for example: TUT, test, 1.0). The corresponding directory will update as you fill in the VLNV.
      * Confirm the creation of the design by pushing “OK”
* Add a signal generator
  + Select “Plain IP-Xact” under the “IP-XACT Library” section on the left
  + Go to component “sig\_gen” and drag the “1.0” version (component/TUT/ip.hwp.accelerator/sig\_gen/1.0) of the component into the design section in the center in the “Components” column
* Add a port blinker
  + Repeat the previous prodecure for the component “port\_blinker” (component/TUT/ip.hwp.accelerator/port\_blinker/1.0)
* Add ports for the design
  + Select the green port tool in “Diagram Tools” in the toolbar
  + Add three ports to the left “IO” area and one to the right
* Add connections to the design
  + Select the blue connection port tool in “Diagram Tools” in the toolbar
  + Connect either component’s clk signal to a blank port on the left
    - Choose copy when the dialog appears
  + Connect the rest clk and rst\_n ports in the same manner
  + Connect the two signal\_gen\_if ports together
  + Finally connect the port\_out to the blank IO port on the right
* Now you should have a design resembling the figure below



* Lock your design by pushing the lock sign under “Protection” on the toolbar
* Add Quartus pin map
  + Select “Plain IP-Xact” under the “IP-XACT Library” section on the left
    - Search for your design under component
    - Right-click on your design and choose “Open Component”
    - Un-lock the component by pushing the lock sign under “Protection” on the toolbar
    - Add a new file set
      * Select “File sets” and push the green plus sign
      * Type “quartusFiles” as the name
      * Select “File sets/quartusFiles/Files” and push the green plus sign
      * Select the file “altera\_de\_II\_demo.qsf” under the directory “trunk/TUT/soc/altera\_de\_II\_demo/1.0/quartus/atom\_netlist”
      * Specify a new file type
        + Push the “Add new” button in the “General settings” dialog under Specified file types
        + Write “quartusPinmap” as the name
* Lock the component
* Generate VHDL and Quartus project files
  + Select your design and unlock it
  + Push the green VHDL arrow under “HW Design” on the toolbar
  + Create a subdirectory “vhd” and go in to it and push “Save”
  + Create a subdirectory “quartus” and go in to it and push “Open”
* Synthesise and program the design under Quartus
  + The Quartus project is generated under the quartus sub-directory
  + Change the device setting of the project to Cyclone II EP2C35F672C6
  + Synthesize and program the project

# Launching Kactus 2

PCI-Express (PCIe) to Heterogeneous IP Block Interconnect (HIBI) is an adapter component which with the help of Altera’s PCIe controller interfaces a PCIe interconnect to a HIBI bus.



Figure - PCIe to HIBI adapter interfacing PCIe phy to HIBI phy

### Ports and signals

The interface signals can be divided into the PCIe controller side and to the HIBI side. The PCIe controller side has four different signal groups which include the Avalon ST RX, Avalon ST TX, PCIe IRQ and the LM interface. The HIBI side has also four signal groups which include the HIBI RX, HIBI TX, HIBI message RX and HIBI message TX.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal group | Signal | Width [bits] | Dir. | Meaning |
| Reset | rst\_n | 1 | i | Active low reset |
| Clocks | clk | 1 | i | Clock, active on rising edge. |
| HIBI RX | hibi\_addr\_in | 32 | i | HIBI read address |
| hibi\_data\_in | 32 | i | HIBI read data |
| hibi\_comm\_in | 3 | i | HIBI read comm type (write, read req…) |
| hibi\_empty\_in | 1 | i | HIBI read empty |
| hibi\_re\_out | 1 | o | HIBI read enable |
| HIBI TX | hibi\_addr\_out | 32 | o | HIBI write address |
| hibi\_data\_out | 32 | o | HIBI write data |
| hibi\_comm\_out | 3 | o | HIBI write comm type |
| hibi\_full\_in | 1 | i | HIBI write full |
| hibi\_we\_out | 1 | o | HIBI write out |
| HIBI message RX | hibi\_msg\_addr\_in | 32 | i | HIBI message read address |
| hibi\_msg\_data\_in | 32 | i | HIBI message read data |
| hibi\_msg\_comm\_in | 3 | i | HIBI message read comm type |
| hibi\_msg\_empty\_in | 1 | i | HIBI message read empty |
| hibi\_msg\_re\_out | 1 | o | HIBI message read enabe |
| HIBI message TX | hibi\_msg\_addr\_out | 32 | o | HIBI message write address |
| hibi\_msg\_data\_out | 32 | o | HIBI message write data |
| hibi\_msg\_comm\_out | 3 | o | HIBI message write comm type |
| hibi\_msg\_full\_in | 1 | i | HIBI message write full |
| hibi\_msg\_we\_out | 1 | o | HIBI message write out |
| Avalon ST RX | Rx\_St\_Data\_i | 128 | i | Avalon ST receive data |
| Rx\_St\_Valid\_i | 1 | i | Avalon ST receive data valid |
| Rx\_St\_Sop\_i | 1 | i | Avalon ST receive start of packet |
| Rx\_St\_Eop\_i | 1 | i | Avalon ST receive end of packet |
| Rx\_St\_Bardec\_i | 8 | i | Avalon ST receive bar decoded |
| Rx\_St\_Be\_i | 16 | i | Avalon ST receive byte enable |
| Rx\_St\_Ready\_o | 1 | o | Avalon ST receive ready |
| Rx\_St\_Mask\_o | 1 | o | Avalon ST receive mask |
| Avalon ST TX | Tx\_St\_Sop\_o | 1 | o | Avalon ST transmit start of packet |
| Tx\_St\_Eop\_o | 1 | o | Avalon ST transmit end of packet |
| Tx\_St\_Valid\_o | 1 | o | Avalon ST transmit data valid |
| Tx\_St\_Data\_o | 128 | o | Avalon ST transmit data |
| Tx\_St\_Ready\_i | 1 | i | Avalon ST transmit ready |
| TxCred\_i | 36 | i | Avalon ST transmit credit |
| PCIe IRQ | app\_msi\_req | 1 | o | Interrupt request |
| app\_msi\_ack | 1 | i | Interrupt acknowledge |
| app\_msi\_tc | 3 | o |  |
| app\_msi\_num | 5 | o | Interrupt number |
| pex\_msi\_num | 5 | o |  |
| app\_int\_sts | 1 | o |  |
| app\_int\_ack | 1 | i |  |
| LMI | lmi\_data\_in | 32 | i | LMI read data |
| lmi\_re\_out | 1 | o | LMI read enable |
| lmi\_we\_out | 1 | o | LMI write enable |
| lmi\_ack\_in | 1 | i | LMI acknowledge |
| lmi\_addr\_out | 12 | o | LMI address |
| lmi\_data\_out | 32 | o | LMI write data |

Table 4- PCIe to HIBI ports