*HUNT ENGINEERING*
**Chestnut Court, Burton Row,**
**Brent Knoll, Somerset, TA9 4BP, UK**
**Tel: (+44) (0)1278 760188,**
**Fax: (+44) (0)1278 760199,**
**Email: sales@hunteng.co.uk**
**http://www.hunteng.co.uk**
**http:www.hunt-dsp.com**

# *HUNT ENGINEERING*

# *HERON-IO2*

***HERON Module with 200K/1M gate FPGA***
***and 2 channels of 125Mhz 12 bit A/D***
***and 2 channels of 125Mhz 14 bit D/A***

# *USER MANUAL*

***Hardware Rev C***
***Document Rev H***
***P.Warnes/T.Hollis 30/06/06***

## COPYRIGHT

## WARRANTIES LIABILITY and INDEMNITIES

HUNT ENGINEERING warrants the hardware to be free from defects in the material and workmanship for 12 months from the date of purchase. Product returned under the terms of the warranty must be returned carriage paid to the main offices of HUNT ENGINEERING situated at BRENT KNOLL Somerset UK, the product will be repaired or replaced at the discretion of HUNT ENGINEERING.

> Exclusions - If HUNT ENGINEERING decides that there is any evidence of electrical or mechanical abuse to the hardware, then the customer shall have no recourse to HUNT ENGINEERING or its agents. In such circumstances HUNT ENGINEERING may at its discretion offer to repair the hardware and charge for that repair.
>
> Limitations of Liability - HUNT ENGINEERING makes no warranty as to the fitness of the product for any particular purpose. In no event shall HUNT ENGINEERING'S liability related to the product exceed the purchase fee actually paid by you for the product. Neither HUNT ENGINEERING nor its suppliers shall in any event be liable for any indirect, consequential or financial damages caused by the delivery, use or performance of this product.

Because some states do not allow the exclusion or limitation of incidental or consequential damages or limitation on how long an implied warranty lasts, the above limitations may not apply to you.

## TECHNICAL SUPPORT

Technical support for HUNT ENGINEERING products should first be obtained from the comprehensive Support section http://www.hunteng.co.uk/support/index.htm on the HUNT ENGINEERING web site. This includes FAQs, latest product, software and documentation updates etc. Or contact your local supplier - if you are unsure of details please refer to http://www.hunteng.co.uk for the list of current re-sellers.

HUNT ENGINEERING technical support can be contacted by emailing support@hunteng.demon.co.uk, calling the direct support telephone number +44 (0)1278 760775, or by calling the general number +44 (0)1278 760188 and choosing the technical support option.

# TABLE OF CONTENTS

The HERON module is a module defined by HUNT ENGINEERING to address the needs of our customers for real-time DSP systems. The HERON module is defined both mechanically and electrically by a separate HERON module specification that is available from the HUNT ENGINEERING CD, via the user manual section from the CD browser, or online from http://www.hunteng.co.uk and going to the application note section under the user area.

The HERON module specification also defines the features that a HERON module carrier must provide. HERON stands for Hunt Engineering ResOurce Node, which tries to make it clear that the module is not for a particular processor, or I/O task, but is intended to be a module definition that allows "nodes" in a system to be interconnected and controlled whatever their function. In this respect it is not like the TIM-40 specification which was specific to the 'C4x DSP.

As the HERON-IO2 was developed, HUNT ENGINEERING have already developed HERON modules carriers like the HEPC9, HERON processor modules (that carry various other members of the TMS320C6000 family of DSP processors from TI), and are working on new HERON-IO modules. In addition to these modules, the HERON specification is a super-set of the pre-existing HUNT ENGINEERING GDIO module, so the GDIO modules from our C4x product range can also be used in HERON systems.

The HERON module connects to the carrier board through several standard interfaces.

- The first is a FIFO input interface, and a FIFO output interface. This is to be used for the main inter-node communications. (It is usually also used for connection to the HOST computer if any).
- The second is an asynchronous interface that allows registers etc to be configured from a HERON module. This is intended for configuring communication systems, or perhaps to control some function specific peripherals on the carrier board.
- The third is a JTAG (IEEE 1149.1) interface for running processor debug tools.
- The Fourth is the HERON Serial Bus, used for configuration messages.
- The last is the general control such as reset, power etc.

HUNT ENGINEERING defined the HERON modules in conjunction with HEART – the Hunt Engineering Architecture using Ring Technology. This is a common architecture that we will adopt for our HERON carriers that provides good real time features such as low latency and high bandwidth, along with software reconfigurability of the communication system, multicast, multiple board support etc., etc.

However, it is not a requirement of a HERON module carrier that it implements such features. In fact our customers could develop their own module carrier and add our HERON modules to it. Conversely our customers could develop application specific HERON modules themselves and add them to our systems.

The HERON-IO2 is HERON module that has 2 channels of 125MHz 12 bit A/D and 2 channels of 125Mhz 14 bit D/A and some General Purpose Digital I/O. This means the HERON-IO2 will be used mainly as a flexible Analog I/O module, but one that can optionally be programmed to perform hardware signal processing on that I/O.

The HERON-IO2V provides a 1M gate FPGA from the Xilinx Virtex II family. This is a high performance part that has specific optimisations for Digital Signal Processing that can be loaded with bit-streams provided by HUNT ENGINEERING or developed by the user.

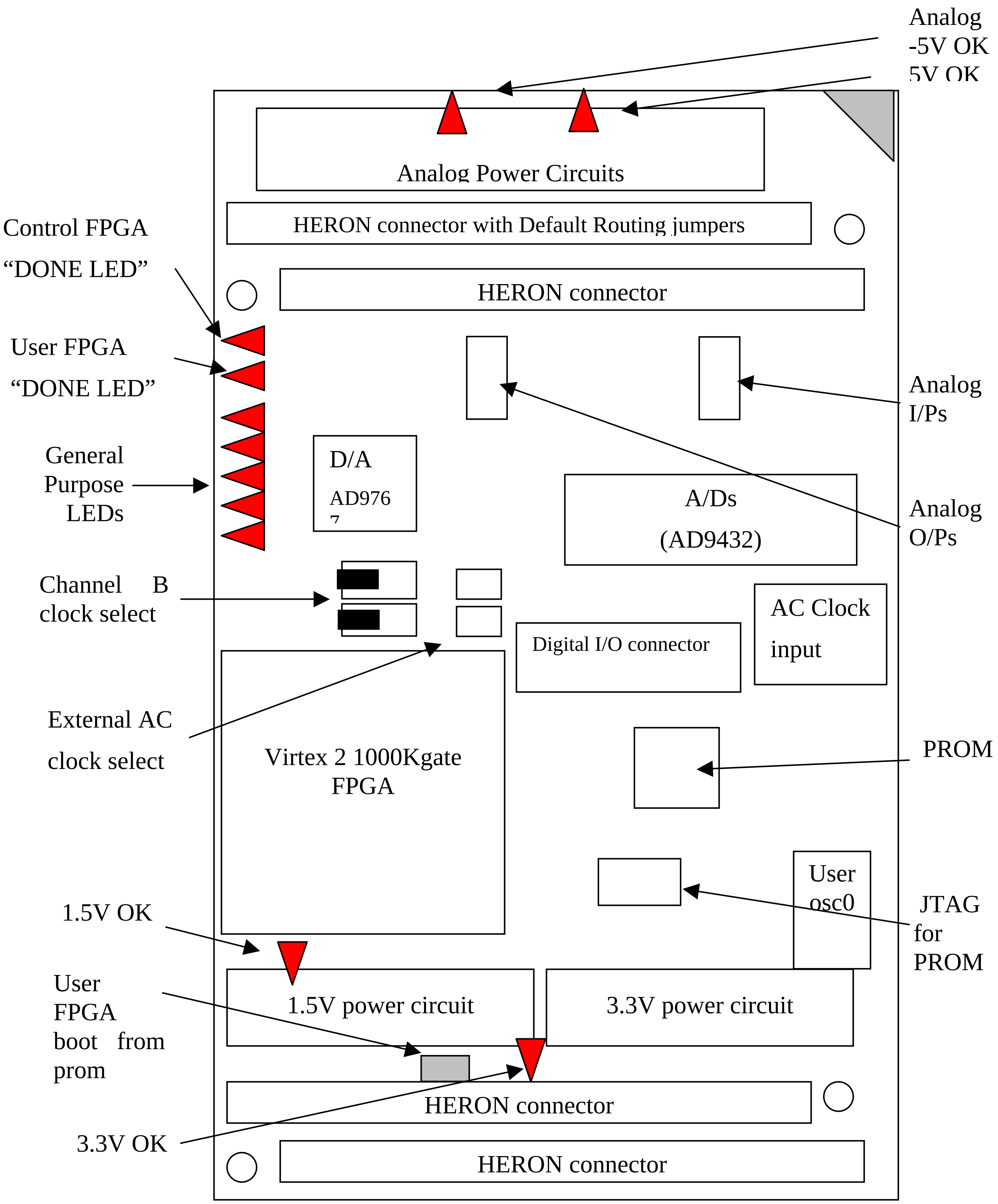
The HERON-IO2 connects all of the HERON module signals, except the JTAG, to the FPGA, allowing flexible use of the module's resources.

The HERON-IO2 provides some of the FPGA I/Os connected to a digital I/O connector on the module. This allows the FPGA to be configured for a variety of I/Os. There are also configurable components that can provide RS232, RS485, and even differential ECL (allowing the possibility of an FPDP interface).

The HERON-IO2 uses the HERON module's serial bus to download configuration bit streams into the FPGA, allowing the user to configure it with standard functions provided by HUNT ENGINEERING or functions that they have developed themselves using the Xilinx development tools.

Programmers of the FPGA can also use more comprehensive development packages for the FPGA such as the Foundation ISE 5 from Xilinx. These tools require a license fee to be paid to Xilinx.

There is also the possibility for the module to configure the FPGA part from a JTAG programmable FLASH ROM. This is intended to simplify the deployment of systems after the FPGA functions have been fully developed.


Analog
-5V OK
5V OK
Analog Power Circuits
HERON connector with Default Routing jumpers
Control FPGA
"DONE LED"
HERON connector
User FPGA
"DONE LED"
Analog
I/Ps
General
Purpose
LEDs
D/A
AD976
7
A/Ds
(AD9432)
Analog
O/Ps
Channel B
clock select
AC Clock
input
Digital I/O connector
External AC
clock select
Virtex 2 1000Kgate
FPGA
PROM
User
osc0
JTAG
for
PROM
1.5V OK
1.5V power circuit
3.3V power circuit
User
FPGA
boot from
prom
HERON connector
3.3V OK
HERON connector

The HERON-IO2 is a module that plugs into a HERON module carrier.

The HERON-IO2 should be fitted to the carrier card along with any other modules that your system has and their retaining nuts fitted (see a later section of this manual for details).

The Default routing jumpers must be set correctly for the system. For most systems the FPGA based modules will not require any default routing jumpers to be fitted. This will allow the FPGA to access whatever FIFO is needs to, and will rely on the FIFO being connected to the right place by the carrier configuration. (See a later section for details on default routing jumpers).

There are three user configurable jumpers on the HERON-IO2, but most applications will not need to change them from their default (shipping) condition as shown in the earlier drawing. This shipping condition is to select a common clock for the A/Ds sourced from the FPGA unless we have provided a special configuration for you. If you think that these do need to be changed, then see the later sections in this manual for details.

The HERON-IO2, following reset, will enter a state where it can be interrogated and programmed using the HERON module's serial bus. It is addressed according to the Carrier number and the slot number of the HERON slot that it is fitted to.

The FPGA configuration data will have been generated using the Xilinx development tools. HUNT ENGINEERING provides examples for the HERON-FPGA modules in the correct format for use with the Xilinx Foundation software. HUNT ENGINEERING also provides software for the Host PC that will allow the output files from the Foundation software to be loaded onto a HERON-FPGA module.

Follow the "Starting your FPGA development" tutorial from the Getting started section of the HUNT ENGINEERING CD, and then the examples that are specific to the HERON-IO2.

It could be possible to use the HERON-IO2 as an I/O module using one of the example bit streams. In this case it is not necessary to be concerned how to program the FPGA – simply load the example bit stream and use it.

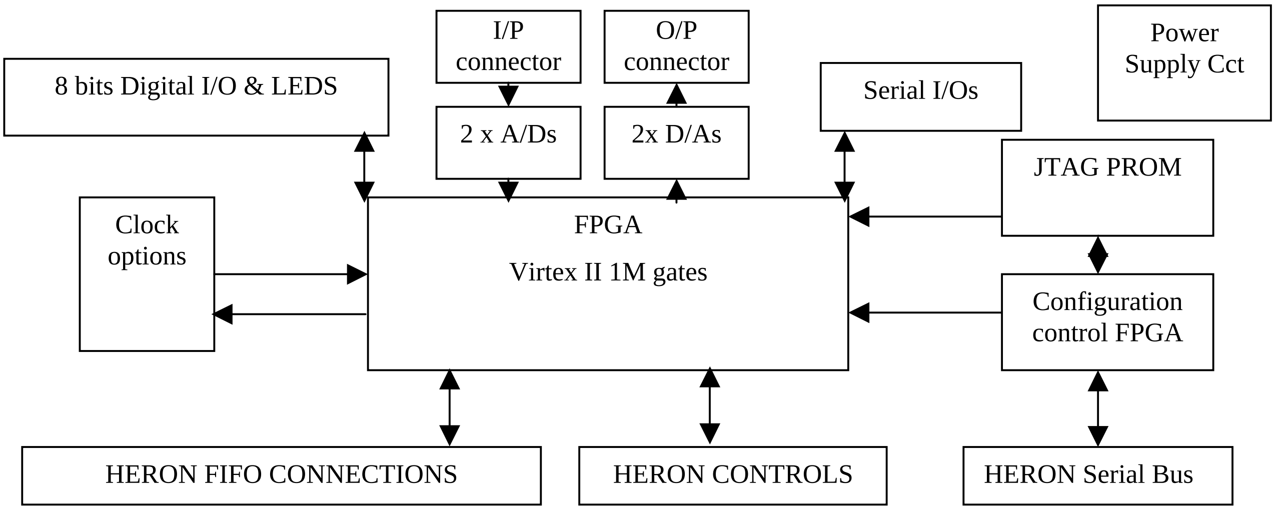
## Standard Intellectual Property (IP)

HUNT ENGINEERING provides examples for the HERON-IO2 that perform different functions. It is possible to use these standard configurations directly if they fit your needs.

It could be possible to request a new standard example from HUNT ENGINEERING, which could avoid the need to purchase and learn how to use the FPGA development tools. Depending on the complexity of your request HUNT ENGINEERING may choose not to offer it, or to charge for it.

New IP for the HERON-IO2 will be posted on the HUNT ENGINEERING web site in the user area whenever it becomes available. HERON-IO2 users can then take advantage of that IP free of charge.

This section describes the features of the HERON-IO2 and why they are provided.



## BLOCK diagram

## Serial Configuration of the User FPGA

The HERON-IO2 usually has the configuration of the User FPGA downloaded using the HERON module’s serial configuration bus. This allows the use of “standard” configurations as supplied on the HUNT ENGINEERING CD, or of user defined configurations without the need to return the module to the factory.

It is imagined that as the “standard” set of functions grows, that they be made available to users of HERON-FPGA and IO modules via the HUNT ENGINEERING web site or CD update requests. Also HUNT ENGINEERING will have the possibility to provide semi-custom configurations for a charge via email.

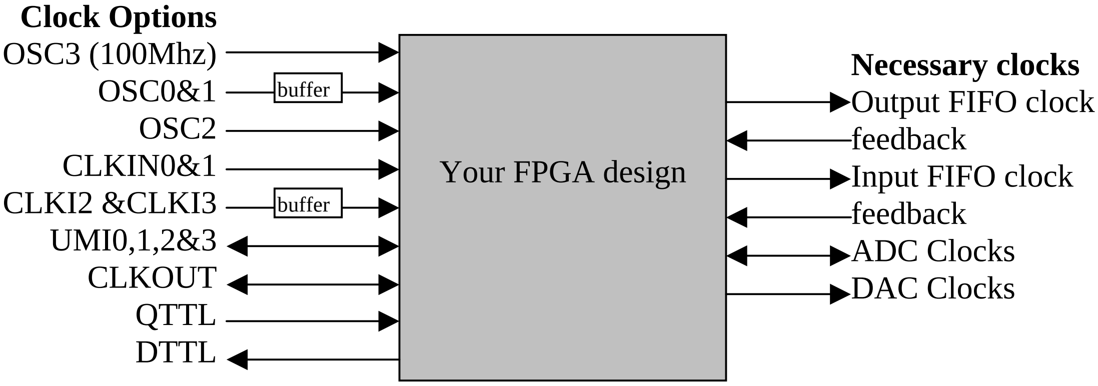
## Boot ROM

As an alternative to the serial configuration download, it is possible to configure the FPGA from a Flash based configuration PROM. It is an advantage when the DSP system will be ROM booted, as it removes the need for the DSP FLASH ROM to store the configuration for the FPGA too. However, if a system is being deployed with a host machine such as a PC, it might be preferable to continue to use the serial configuration method, as this will make in field upgrades and bug fixes simpler to deploy.

The PROMS fitted to the HERON-IO2 are Flash based, and can be programmed (and re-programmed) using a Xilinx JTAG cable (such as Xilinx parallel cable 3 or 4).

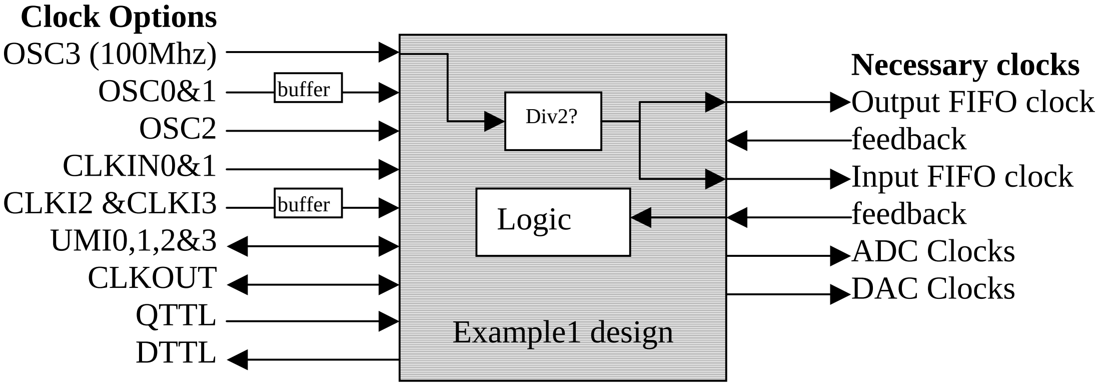
## Clocking of the FPGA

The Xilinx FPGA used on the HERON-IO2 does not have a single clock input, but rather it can use any one of its pins to provide a clock input. This means you can have many sources of clocks, each of which can be used inside your FPGA design. You can even divide these clocks using flip flops, or even multiply using digital clock manager components.



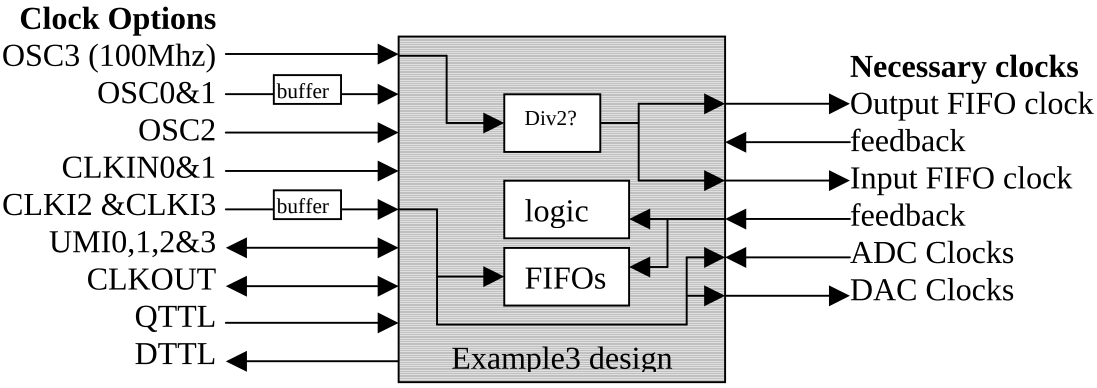
The simplest way to manage your FPGA design is to use just one clock throughout your design. However the FPGA must drive both of the HERON FIFO clocks, at a frequency that is suitable for your module carrier (see the documentation for your module carrier for details of its restrictions). The FPGA must also drive the A/D clocks, and the D/A clocks. The frequency for these might be limited by the components, or by the needs of your signal processing. The needs for these clocks can only be determined by looking carefully at the needs of your system.

The A/D sample clock can be sourced externally through the 'AC CLK' input, this also provides another clock source for the FPGA. The A/C clock input can be used, either with a low level sinusoidal clock input or an LVTTL squarewave input. When the correct jumpers are fitted this clock drives the A/D directly with a 350ps maximum delay, allowing a low phase error and no additional jitter. In that case ADC clock becomes an input to the FPGA for use in the associated logic.



If these clocks cannot be the same, then the next best situation is to have one clock derived from the other. In that way the relationship between the clock edges will be known.

The most difficult case for your FPGA design is to have many clocks from different sources that are all used in the same design. Then you must carefully manage signals that cross from one clock “domain” to another. This can be handled by FIFOS, or by multiple registering to prevent metastability problems. Refer to texts on digital design to understand these issues.



Example3 for the HERON-IO2 uses three separate clock sources, one for the ADCs (CLKI2) one for the DACs (CLKI3) and another for the FIFOS and logic (OSC3). This example uses FIFOS to pass data from one clock domain to the other. On the io2v2, i.e. modules after S/N 058 the ADC clock is taken from the AC clock input rather than the CLKI2.

The HERON-IO2 provides a highly flexible set of choices for the clocking of the FPGA, D/As and A/Ds.

The HERON-IO2 has a socket for a user oscillator. It has a further 3 locations for Surface mount oscillators to be fitted for designs that require all 4 user oscillators to be used.

Default shipping state is to have a 100Mhz oscillator fitted to one of the surface mount sites – driving 100Mhz on UserOsc3. This is a standard commercial oscillator module, that is +-100ppm accuracy. If you require higher accuracy for your analogue clocks then you should use one of the other clock sources.

There is a “Digital I/O” connector which also provides for up to 4 external clock inputs, and a clock output. Also the UMI pins on the module connector could be used as a clock input, if another module in the system is programmed to drive that clock onto the UMI connection.

## HERON FIFOS

The HERON module can access up to 6 input FIFOs and up to another 6 output FIFOs. Actually it is most likely that a carrier board will not implement all 12 FIFO interfaces. Each FIFO interface is the same as the others, using common clocks and data busses.

The input and output interfaces are separate though, allowing data to be read and written at the same time by a module like the HERON-IO2.

While it is possible to read one FIFO and write another FIFO at the same time, the use of shared pins means no more than one can be written or read at the same time.

For each interface (input or output) there is a FIFO clock that must be a constant frequency, and running constantly. There may be some minimum and maximum frequency requirements for a particular Module Carrier card that the designer of the FPGA contents must be sure to comply with. This is because the FIFO clocks are generated by the FPGA, probably based on one of the clock inputs to the FPGA .

Each FIFO interface has a separate "enable" signal that is used to indicate which FIFO is accessed using the clock edge.

### Input FIFOs

The input FIFOs use a common data bus that is driven onto the HERON module. It is important to ensure that no more than one of the FIFOs are read at the same time, but more importantly that no more than one has its output enable selected.

By properly asserting the "read enable" and "output enable" signals relative to the clock the FPGA can access the FIFO of its choice at a rate up to one 32 bit word per clock cycle.

For the timing of those signals refer to the HERON module specification.

Each input FIFO interface provides Flags that indicate the state of the FIFO. An empty flag shows that there is no data to be read, an almost empty flag shows that there are at least 4 words left. While the almost flag is not asserted accesses can be made on every clock, but after it is asserted, it is better to make one access only, then check the empty flag on the next clock, before deciding if another access is possible.

### Output FIFOs

The output FIFOs use a common data bus that is driven by the HERON module. It is important to ensure that no more than one of the FIFOs is written at the same time – unless that is required by your system.

By properly asserting the "write enable" signals relative to the clock, the FPGA can access the FIFO of its choice at a rate up to one 32-bit word per clock cycle.

For the timing of those signals refer to the HERON module specification.

Each output FIFO interface provides Flags that indicate the state of the FIFO. A full flag shows that there is no room left to write, an almost full flag shows that there are at least 4 words of space left. While the almost flag is not asserted accesses can be made on every clock, but after it is asserted, it is better to make one access only, then check the full flag on the next clock, before deciding if another access is possible.

### FIFO clocks

The FIFO clocks are provided by the FPGA, but are buffered externally using an LVT245 buffer that is able to provide the drive current required on these signals. To enable circuitry internal to the FPGA to be designed to use the actual clock that is applied to the FIFO, the buffered FIFO clock signals are connected to some GCLK inputs. This allows DLLs to be used to provide a clock internal to the FPGA that has the same phase as that applied to the FIFOs on the carrier board.

## Analog I/O

The HERON-IO2 connects the FPGA to two 12 bit A/D chips that can sample at rates between 1 and 125Mhz. This allows the User FPGA program to sample the A/D data, process it, store it or pass it on to another HERON module as the user's system requires.

The HERON-IO2 also connects the FPGA to two 14 bit D/As that can be clocked at up to 125Mhz.

### A/D clocking

The A/Ds are clocked by the FPGA to allow control of the sampling rate, which can be derived from any of the User Oscillator sockets, external clock inputs or even HERON UMI pins. Two of the GCLK pins from the FPGA are connected to user I/Os that could be driven with the same clock as the A/Ds so that DLLs can be used to synchronise clocks within the FPGA to the A/D sample clock with known phase. The management if this is taken care of in the Hardware Interface Layer vhdl provided.

The FPGA drives directly the sample clock input of the A/D components, with no buffer delays.

Each A/D has a separate clock input so that they can be driven with different frequencies or phases according to the user FPGA configuration. For applications that require them to be driven with exactly the same phase of clock there is a jumper that can be fitted to use the same clock for both channels.

There is the added option of clocking either one or both the A/D's directly from the external "AC Clock" input, which is also fed to the FPGA to use for the interfacing logic.

The reason for this "option" is so an external clock can be used for the A/Ds without the FPGA introducing any additional jitter. The phase delay is determined by the buffer delay which is a maximum 350ps. Such a short delay cannot be achieved if the clock source is fed through the FPGA.

The lower frequency limit of the "AC clock" input for clocking the A/D converters is approximately 4 MHz when driving the input with a 0 dBm sine wave.

### D/A clocking

The D/As are clocked by the FPGA to allow control of the output rate, which can be derived from any of the User Oscillator sockets, external clock inputs or even HERON UMI pins.

The FPGA drives directly the sample clock inputs of the D/A components, with no buffer delays.

Each D/A has a separate clock input so that they can be driven with different frequencies or phases according to the user FPGA configuration. Because the D/As provide continuous analogue outputs, a small phase is not considered important, so there is no option to use *exactly* the same clock.

## Digital I/O

The HERON-IO2 connects 8 of the FPGA's I/O pins to connectors. This allows them to be configured as digital Inputs and Outputs as chosen by the user's FPGA program.

On the HERON-IO2V, some module specific LVTTL inputs are connected to bank 1 that require a 3.3V Vcco. This precludes the setting of any other value for the Vcco1, so the formats supported by the Digital I/O connectors are those supported by the Virtex-II FPGA with a Vcco of 3.3V

## Module and Carrier ID

The HERON specification assigns pins on the HERON module that give a HERON module access to the carrier ID of the carrier that it is plugged into, and a unique HERON slot identifier.

These IDs are used by the configuration FPGA so that the module is addressed on Heron Serial Bus (HSB) using this information. These signals are also connected to the user FPGA so a user program can use them if required.

## General Purpose LEDs

There are some LEDs on the HERON-IO2 that are connected via a buffer to some of the FPGA I/O pins. There are five such LEDs, which can be used by the FPGA program to indicate various states of operation.

## Done LEDs

There are two Done LEDs, that are illuminated if the relevant FPGA is not configured.

LED "DONE" is connected to the User FPGA

LED "CNT DONE" is connected to the Control FPGA.

This means that the "CNT DONE" LED should flash at power on, and then go out showing that the control FPGA is ready to accept a configuration stream for the User FPGA.

After downloading a bitstream to the user FPGA, LED "DONE" should also go out.

## Serial Ports

There is a connector on the HERON-IO2 that provides the opportunity to use a variety of serial port configurations. There is a MAX3160 "protocol converter" chip provided external to the FPGA, which can provide: -

| | |
|---|---|
| 1 channel 4 wire RS232 | With RTS and CTS |
| 2 channels of 2 wire RS232 | Without RTS and CTS |
| 1 channel of 4 wire RS485 | One pair in each direction |
| 2 channels of 2 wire RS485 | Two bi-directional pairs |

In addition there is a Differential ECL transceiver chip provided that gives one differential ECL input and one differential ECL output, such as is used by FPDP.

The components are purely voltage level converters, and any UART logic must be implemented in the user FPGA.

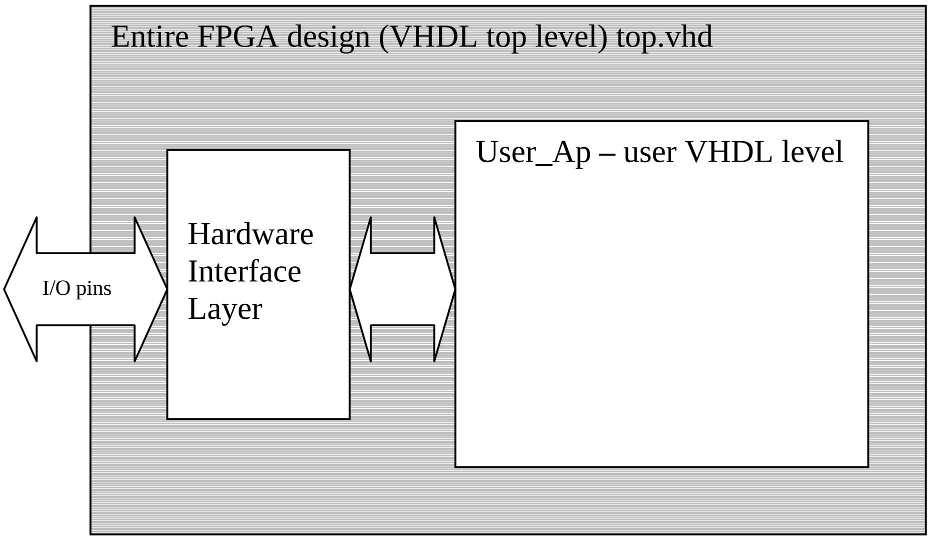
Note that Version 1 modules returned a module type of “io2v1” when interrogated over HSB. Because of the clocking changes that have been made for the version 2 boards these modules return “io2v2”. This allows the development tools to differentiate between the modules and use the correct examples. There are two separate directories that contain the support for the different types, called io2v1 and io2v2.

HUNT ENGINEERING provide a comprehensive VHDL support package for the HERON-IO2.

This package consists of a VHDL “top level”, with corresponding user constraints file, VHDL sources and simulation files for the Hardware Interface Layer, and User VHDL files as part of the examples.

The Hardware Interface Layer correctly interfaces with the Module hardware, while the top level (top.vhd) defines all inputs and outputs from the FPGA on your module. Users should not edit these files unless a special digital I/O format is required – see the later section “Digital I/O from the FPGA”.

The file user_ap.vhd is where you will make your design for the FPGA, using the simplified interfaces provided by the Hardware Interface layer.


Entire FPGA design (VHDL top level) top.vhd
User_Ap – user VHDL level
Hardware Interface Layer
I/O pins

Organisation of VHDL support for FPGA modules.

After synthesising your design, you will use the Place and Route tools from Xilinx (either as part of your ISE package, of from the Xilinx Alliance tools). These tools will use the User Constraints File (.ucf) to correctly define the correct pins and timing parameters. You will need to minimally edit this file to have the timing constraints that you need, but the file provided means you do not need to enter the pin out constraints at all.

It is expected that every user will start by following the Getting Started example, Example1, which is supplied on the HUNT ENGINEERING CD. By working through the Getting Started example you will be able to see how the User FPGA is configured, how a simple example can be built, and a new bitstream generated.

In this way, you can use example1 to check your understanding of how the module works, and you can also use the example as a sanity check that your hardware is functioning correctly.

## Working through Example 1

All HERON modules that have FPGAs have an "Example1" provided for them. It is a simple example that connects data from the input FIFO interface to the output FIFO interface, and also exercises the HSB interface.

This example is fully described in the "Starting your FPGA development" tutorial on the HUNT ENGINEERING CD.

The tutorial works through running the example and then modifying and re-building it. The tutorial assumes that you are using the latest version of the ISE Foundation tool-set from Xilinx. If you are using a different version of ISE Foundation, you will simply need to convert the project as described in the application note provided by HUNT ENGINEERING titled 'Using Different Versions of ISE'. There is also an application note on the HUNT ENGINEERING CD that describes using design flows that are different from ISE, titled 'Using VHDL tools other than ISE'.

The example is quite simple but demonstrates the use of the interfaces found in the Hardware Interface Layer supplied with the module. The example is supplied in two ways. Firstly, there is a ready-to-load bitstream, supplied in the Hunt Compressed Bitstream file format, or '.hcb' format. This is the file format used by the HUNT ENGINEERING configuration tool. Secondly, there is an example1 project supplied for ISE, enabling the design to be rebuilt and a new bitstream downloaded.

The bitstream file is provided to allow you to load the example1 onto the hardware without having to re-build it. This is a useful confidence check to see if any problems you are experiencing are due to changes you have made, or the way you have built the design. If the bitstream from the CD fails to behave then the problem is more fundamental.

To make things easier, we have created the proper ISE project files for the examples.

Using these projects will allow you to run the complete design flow, from RTL-VHDL source files to the proper bitstream, ready to download on your Heron FPGA board.

No special skills are required to do this.

However, if you want to write your own code and start designing your own application, you must make sure that you have acquired the proper level of expertise in:
*VHDL language,

*Digital Design

*Xilinx FPGAs

*ISE environment and design flow

Proper training courses exist which can help you acquire quickly the required skills and techniques. Search locally for courses in your local language.

## Preparing ISE

Before beginning work with Example1 you will need to make sure ISE is properly installed. In addition, you should ensure that you have downloaded the latest service pack from the Xilinx website for the version of ISE you are using.

## Copying the examples from the HUNT ENGINEERING CD

On the HUNT ENGINEERING CD, under the directory "fpga" you can find directories for each module type. In the case of the HERON-IO2 the correct directory is "io2v2".

There are two ways that you can copy the files from the CD.

1. The directory tree with the VHDL sources, bit streams etc can be copied directly from the CD to the directory of your choice. In this case there is no need to copy the .zip file, but the files will be copied to your hard drive with the same read only attribute that they have on the CD. In this case all files in the "examplex" directories need to be changed to have read/write permissions. It is a good idea to leave the permissions of the common directory set to read only to prevent the accidental modification of these files.
2. To make the process more convenient we have provided the zip file, which is a zipped image of the same tree you can see on the CD. If you "unzip" this archive to a directory of your choice, you will have the file permissions already set correctly.

## Opening the Example1 Project

Let us start with Example1. In the tree that you have just copied from the CD, open the `Example1` sub-directory. You should see some further sub-directories there:

* ISE holds the ISE project files.
* Src holds the application-specific Source files.
* Sim holds the simulation scripts for ModelSim.
* Leo_Syn holds the synthesis scripts for Leonardo Spectrum and Synplify users.
You may ignore this directory in this chapter.

Open the Xilinx ISE Project Navigator. If a project pops up (from a previous run), then close it. Use File → Open project.

Select `Example1\ISE\XXXXX.ise` and click on the `"Open"` button.

After some internal processing, the "Sources window" of the Project Navigator will display the internal hierarchy of the Example1 project.

If you are encountering errors at this stage, you should verify that:

The example files have been correctly copied onto your hard disk, and especially the \Common and Example1\Src directories.

The correct version of ISE has been successfully installed. Be sure to have installed XST VHDL synthesis and the support for the Virtex2 family.

## The Project's functional parameters

Double click on "user_ap1" in the Sources window.
This opens the VHDL colour-coded text editor so that you can see the part of the project where you can enter your own design.

The first code that you will see at the beginning of this file is a VHDL Package named "config" which is used to configure the design files according to the application's requirements. See the next section of this manual for a description of these items

Below the package section, you will see the User_Ap1's VHDL code.

This is where you will insert your own code when you make your own design.

We provide a system which is built in such a way that the user should not need to edit any other file than User_Ap (and the entities that this module instantiates).

**In particular, the user should NOT modify the HE_* files, even when creating new designs for the FPGA.**

## Setting up the Configuration Package

At the top of the file USER_APx.VHD (where x indicates the example number) there are settings that you can change to affect your design (in this case the example). The idea is that settings that are often changed are found here.

### 1. Divide External Clock by 2

The example uses the 100Mhz oscillator that is fitted to Osc3 of the module. It generates the FIFO clock either directly from this 100Mhz, or divides it by 2 to generate a 50Mhz FIFO clock.

Set this parameter to "True" if you want to divide the external clock by two and use this as your main Clock.

If you are using an HEPC9 carrier board, set DIV2_FCLK to "False".

### 2. Fifo Clocks

You must decide whether you will have a single common clock for driving the input and output FIFOs. Normally a design is simpler if the same clock is used for input and output FIFOs, but the module design allows you to use different frequencies or phases if that is more convenient for the design of your system. Whether you use a common clock or separate clocks will affect your design, but it also affects the use of clocks in the Hardware Interface Layer.

Set FCLK_G_DOMAIN to True if you have the same clock driving both FIFOs.
This is the default option for the Examples. If you are unsure, select this choice.

Then, you must know whether your clocks are running slower than 60 MHz or not. This is the frequency that *you* connect to the SRC_FCLK_G in your design.

Set HIGH_FCLK_G to True if your global clock is running at 60 MHz or above. In this case an HF DLL will be used in the FIFO clocks to ensure the proper timing.

Set HIGH_FCLK_G to False otherwise. In this case the HF DLL does not work, and an LF DLL is necessary.

Set FCLK_G_DOMAIN to False is you have a different clock driving each Fifo.
This option should be reserved to advanced users familiar with the management of multiple clock domains systems.

Then, you must know whether each of your clocks are running slower than 60 MHz or not.

These are the frequencies that *you* connect to the SRC_FCLK_RD and SRC_FCLK_WR in your user_ap.

Set HIGH_FCLK_RD to True if your Input Fifo clock is running at 60 MHz or above, so that the HF DLL will be used for the input FIFO clock.

Set HIGH_FCLK_RD to False otherwise, so a LF DLL will be used for the input FIFO clock.

Set HIGH_FCLK_WR to True if your Output Fifo clock is running at 60 MHz or above, so that the HF DLL will be used for the output FIFO clock.

Set HIGH_FCLK_WR to False otherwise, so that a LF DLL will be used for the output FIFO clock.

The Table below summarises the available choices:

| FCLK_G_DOMAIN | HIGH_FCLK_G | HIGH_FCLK_RD | HIGH_FCLK_WR |
|---|---|---|---|
| True | True / False | n.a. | n.a. |
| False | n.a. | True / False | True / False |

In the case of example1, the correct choices are:

| DIV2_FCLK | FCLK_G_DOMAIN | HIGH_FCLK_G | HIGH_FCLK_RD | HIGH_FCLK_WR |
|---|---|---|---|---|
| False | True | True | n.a. | n.a. |

### 3. ADC Clocks

You must decide whether you will have a single common sampling clock for driving both the A and B ADC channels. This is a choice that depends on the requirements of your system. This clock can be the same as the FIFO clock, but it can also be set to a completely different and independent frequency.

Set SCLK_G_DOMAIN to True if you have the same clock driving both ADCs.
This is the default option for the Examples.

Set SCLK _G_DOMAIN to False if you provide two different clocks, one for each ADC.

You must also decide whether you will drive the ADC clock from the FPGA, or accept a clock from the external ADC clock inputs, which can be LVTTL or low level sinewave inputs. When the external clocks are used jumpers must be set to drive the ADC pins and the FPGA pins with the signal that results from the clock input

Set INTERNAL_SCLK_A to be true if you want to use the FPGA to drive the A/D sample clock. In that case the frequency that you want should be driven on the SRC_SCLK_G port. The port SCLK_G should be used by logic connecting to the ADC data.

If you set INTERNAL_SCLK_A to be "false" then any external clock that you connect to the A/D (using the jumpers on the module) will be used by the ADC and input to the FPGA. This clock will be driven onto the port SCLK_G for use by your FPGA logic connected to the ADC interface.

The Table below summarises the available choices:

| SCLK_G_DOMAIN | INTERNAL_SCLK_A | Function |
|---|---|---|
| True | True | FPGA drives common sample clock for both channels<br>Ext CLK jumpers = not fitted<br>A/B jumper = fitted to A/B |
| True | False | External clock used for both channels<br>Ext CLK jumpers = fitted<br>A/B jumper = fitted to A/B |
| False | True | FPGA drives different sample clocks for each channel<br>Ext CLK jumpers = not fitted<br>A/B jumper = fitted to B |
| False | False | External clock used for channel A, and the FPGA drives the sample clock for Channel B<br>Ext CLK jumpers = fitted<br>A/B jumper = fitted to B |

In the case of example1 the ADCs are not used, so the setting of these parameters is not necessary.

### 4. DAC Clocks

The timing requirements of the DAC components inputs are not as strict as the ADC input timings. Hence a DLL is never necessary, and so there are no items in the configuration package regarding the DAC clocks.

## User Timing Constraints

As with all FPGA designs it is necessary to apply some timing constraints to the design to ensure that the tools generate a design that will operate at the frequency that you require. Example1 has these defined in the .ucf file.

Although you can use the configuration package to set the frequency of the FIFO clocks to be 50Mhz, you can still use the stricter timing constraints needed when those clocks run at 100Mhz. So in the case of example1 you do not need to change any of the timing constraints. When you make changes to the design however you may find that you introduce more clock nets that need to be added to the ucf file. In some cases you may find that the tools are unable to achieve your desired clock frequency and then (if you are using an HEPC8) you should change the constraints to reflect you true needs.

For more details on Timing Constraints please refer to the Xilinx tools documentation.

## Creating the Bitstream for Example1

Once the project has been opened as described above:

1. In the "Sources in project" window (Project Navigator), highlight (*single*-click on) the entity 'top' ("..\..\Common\top.vhd").
   This is extremely important! Otherwise, nothing will work!

2. Double-click on the "Generate Programming File" item located in the "Processes for Current Source".

   This will trigger the following activity :

   Complete synthesis, using all the project's source files.
   Since warnings are generated at this stage, you should see a yellow exclamation mark appear besides the "Synthesize" item in the Processes window.

   Complete Implementation :
   - "Translation"
   - Mapping
   - Placing
   - Routing

3. Creation of the bitstream.
   Note that this stage does run a DRC check. Which can potentially detect anomalies created by the Place and route phase.

   When the processing ends, the proper bitstream file, with extension ".rbt" can be found in the project directory. This file MUST be called top.rbt. If it is not then you have synthesised a small part of your design because you did not properly highlight top.vhd in step1.

4. In the "Pad Report" verify a few pins from the busses, like :
   LED(0) = H3, LED(1) = J1, LED(2) = L5, LED(3) = J5, etc... To do this you need to open "implement design" in the processes window, then open "Place and Route". Then double click on the pad report to open it
   If you see different assignments, STOP HERE, and verify the UCF file selected for the project.

You can download this file on your FPGA board and see how it works. See a later section of this manual.

Note that the user_ap level includes a very large counter that divides the main system clock and drives the LED #4. It is then obvious to see if the part has been properly programmed and downloaded: the LED should flash. The hardware will probably require a reset after configuration before the LED starts to flash.

If the LED does not flash, we recommend that you shut down the PC or reprogram the device using a "safe" bitstream. Otherwise, some electrical conflicts may happen (see below).

Possible causes for the device failure to operate are :

1. Wrong (or no) UCF file.
   This happens (for example) if you select the XST-version of the UCF with a Leonardo Spectrum (or Synplify) synthesis. The pin assignment for all the vectors (busses) will be ignored, and these pins will be distributed in a quasi-random fashion!

2. Wrong parameters in the CONFIG package.

3. Design Error.

If nothing seems obvious, rerun the confidence tests, then return to the original example 1.

## Simulating the complete design

To generate the bitstream as above, you did not need to do any simulation. However, if you start modifying the provided examples and add your own code, verification can soon became an important issue.

If you need to simulate your design, you will need to install a VHDL simulator such as ModelSim (available in Xilinx Edition, Personal Edition, or Special Edition).

The example projects provided on the HUNT ENGINEERING CD include simulation files that provide a starting point for simulating your own design. If you wish to work through the simulation examples provide, please read the document 'Simulating HERON FPGA Designs'.

# Making your own FPGA Design

The actual contents of the User FPGA on the HERON-IO2 are generated by the user. While making this development requires some knowledge of Digital Design techniques, it is made quite simple by the development environment that you use.

We recommend the Foundation ISE series software available from Xilinx, because that is what we use at HUNT ENGINEERING, and any examples and libraries we provide are tested in that environment. However there are other tools available from third parties that can also be used. The use of VHDL sources for our Hardware Interface Layer and examples means that virtually any FPGA design tool could be used. Any development tool will eventually use the Xilinx Place and Route tool, where the user constraints file that we supply will ensure that the design is correctly routed for the module. There are application notes on the HUNT ENGINEERING CD that describe how other tools might be used.

The best way to learn how to use your development tools is to follow any tutorials provided with them, or to take up a training course run by their vendors.

ALL NEW PROJECTS SHOULD START FROM ONE OF THE PROVIDED EXAMPLES – that way all of the correct settings are made, and files included. Your design should take place entirely within the User-Ap level, except in the case of needing to change the I/O formats of the Digital I/Os in which case it is necessary to minimally edit the top.vhd file – see a later section in this manual for details.

It is assumed that you are able to use your tools and follow the simplest of Digital Design techniques. HUNT ENGINEERING cannot support you in these things, but are happy to field questions specific to the hardware such as "how could I trigger my A/D from a DSP timer?" or "How can I use the FIFO interface component to do….?".

## User_Ap Interface

This section describes the Interface between the User_Ap central module (or *entity* in VHDL Jargon) and the external Interface hardware. This is the part where you connect your FPGA design to the resources of the module.

In other words :

1. The Clocks system
2. How your application can communicate with the external world : ADC, DAC, HSB interface, and FIFOs.

You need to understand this interface in order to properly connect your processing logic.

The complete FPGA project consists of a Top level in which many sub-modules (*entities*) are placed (*instantiated*) and interconnected. One of these modules is User_AP : ***your*** module. The top-level and the other modules make the system work, but you do not have to understand nor modify them in any way.

Let us see all of the Inputs/Outputs (*Ports*) of your User_Ap module :

Note that the names used for these ports are effectively "reserved" even if the user does not connect to that signal. This means the user must be careful not to re-use the same name for a signal that should not connect to these ports.

| **Port** | **Direction in user_ap** | **Description** |
|---|---|---|
| **GENERAL** | | |
| RESET | In | Asynchronous module reset (active high) |
| CONFIG | In | System config signal (active low) |
| ADDR_FLAGSEL | In | Module input to select the mode of some module pins – see HERON specification |
| BOOTEN | Out | Module output not normally used |
| DIO_EN[0:7] | Out | Digital I/O enables, FPGA output pin driven when low |
| DIO_IN[0:7] | In | Digital I/O in |
| DIO_OUT[0:7] | Out | Digital I/O out |
| UMI_EN[0:3] | Out | Uncommitted Module Interconnect enables, FPGA output driven when low |
| UMI_IN[0:3] | In | Uncommitted Module Interconnects in |
| UMI_OUT[0:3] | Out | Uncommitted Module Interconnects out |
| MID[0:3] | In | Module ID of this module slot |
| CID[0:3] | In | Carrier ID of this carrier |
| UDPRES | Out | Optional reset to system. Drive to ‘1’ if not used. |
| LED[0:4] | Out | 5 x LEDs, can be used for any purpose |
| **CLOCK SOURCES** | | |
| OSC0 | In | External Clock from OSC0 oscillator |
| OSC1 | In | External Clock from OSC1 oscillator |
| OSC2 | In | External Clock from OSC2 oscillator |
| OSC3 | In | External Clock from OSC3 oscillator |
| CLKIN0 | In | Unbuffered External Clock from CLKs connector |
| CLKIN1 | In | Unbuffered External Clock from CLKs connector |
| CLKI2 | In | Buffered External Clock from CLKs connector |
| CLKI3 | In | Buffered External Clock from CLKs |

| | | |
|---|---|---|
| | | connector |
| QTTL | In | Input from external differential LVPECL buffer |
| PECL_CLK | In | Input from external AC clock input |
| **CLOCK OUTPUTS** | | |
| CLKOUT | Out | Unbuffered External Clock to CLKs connector |
| DTTL | Out | output to external differential LVPECL buffer |
| **SERIAL I/OS** | | |
| T1IN | Out | Data driven to RS232/485 chip |
| T2IN | Out | Data driven to RS232/485 chip |
| R1OUT | In | Data input from RS232/485 chip |
| R2OUT | In | Data input from RS232/485 chip |
| RS485_232 | Out | Control signal driven to RS232/485 chip |
| HDPLX | Out | Control signal driven to RS232/485 chip |
| FAST | Out | Control signal driven to RS232/485 chip |
| **FIFO CLOCK INTERFACE** | | |
| FCLK_RD | In | Read FIFO Clock to be used in this module (only when FCLK_G_DOMAIN = FALSE) |
| SRC_FCLK_RD | Out | Input FIFO Clock source for the top level (only when FCLK_G_DOMAIN = FALSE) |
| FCLK_WR | In | Output FIFO Clock to be used in this module (only when FCLK_G_DOMAIN = FALSE) |
| SRC_FCLK_WR | Out | Output FIFO clock source for the top level (only when FCLK_G_DOMAIN = FALSE) |
| FCLK_G | In | Common FIFO clock to be used in this module (only when FCLK_G_DOMAIN = TRUE) |
| SRC_FCLK_G | Out | Common FIFO clock source for the top level (only when FCLK_G_DOMAIN = TRUE) |
| **INPUT FIFOs** | | |
| INFIFO_READ_REQ[5:0] | Out | Input FIFO Read Request |
| INFIFO_DVALID[5:0] | In | Input FIFO Data Valid |
| INFIFO_SINGLE[5:0] | In | Input FIFO Single Word Available |
| INFIFO_BURST[5:0] | In | Input FIFO Burst Possible |
| INFIFO0_D [31:0] | In | Input FIFO 0 Data |

| | | |
|---|---|---|
| INFIFO1_D [31:0] | In | Input FIFO 1 Data |
| INFIFO2_D [31:0] | In | Input FIFO 2 Data |
| INFIFO3_D [31:0] | In | Input FIFO 3 Data |
| INFIFO4_D [31:0] | In | Input FIFO 4 Data |
| INFIFO5_D [31:0] | In | Input FIFO 5 Data |
| **OUTPUT FIFOs** | | |
| OUTFIFO_READY[5:0] | In | Output FIFO Ready for Data |
| OUTFIFO_WRITE[5:0] | Out | Output FIFO Write Control |
| OUTFIFO_D [31:0] | Out | Data written into Output FIFO |
| **HE_USER I/F** | | |
| MSG_CLK | Out | Clock to HE-USER interface logic. |
| MSG _DIN [7:0] | In | Data received from HSB |
| MSG _ADDR [7:0] | In | "address" received from the HSB |
| MSG _WEN | In | Write access from the HSB |
| MSG _REN | In | Read access from the HSB |
| MSG _DONE | In | Message was successfully transmitted (used when initiating HSB messages) |
| MSG _COUNT | In | Counter enable when initiating HSB messages |
| MSG _CLEAR | In | Asynchronous clear for address counter when initiating HSB messages |
| MSG _READY | Out | to acknowledge an access from the HSB |
| MSG _SEND | Out | Message send command (used when initiating HSB messages) |
| MSG _CE | Out | to control speed operation |
| MSG _DOUT [7:0] | Out | Data to be sent to HSB |
| MSG _SEND_ID | Out | Indicates when a byte should be replaced by Own ID (used when initiating HSB messages) |
| MSG _LAST_BYTE | Out | To indicate when the last byte to be sent is presented when initiating HSB messages |
| **ADC INTERFACE** | | |
| SCLK_A | In | ADC – Channel A - clock to be used in this module (only when SCLK_G_DOMAIN = FALSE) |
| SRC_SCLK_A | Out | ADC – Channel A - clock source for the top level (only when SCLK_G_DOMAIN=FALSE) |
| SCLK_B | In | ADC – Channel B - clock to be used in this module (only when SCLK_G_DOMAIN = |

| | | |
|---|---|---|
| | | module (only when SCLK_G_DOMAIN = FALSE) |
| SRC_SCLK_B | Out | ADC – Channel B - clock source for the top level (only when SCLK_G_DOMAIN=FALSE) |
| SCLK_G | In | Common ADC clock to be used in this module (only when SCLK_G_DOMAIN=TRUE) |
| SRC_SCLK_G | Out | Common ADC clock source for the top level (only when SCLK_G_DOMAIN=TRUE) |
| ADC_A [11:0] | In | ADC – Channel A - Input Data |
| OTR_A | In | ADC – Channel A - "Out of Range" flag |
| ADC_B [11:0] | In | ADC – Channel B - Input Data |
| OTR_B | In | ADC – Channel B - "Out of Range" flag |
| **DAC INTERFACE** | | |
| SCLK_DAC_A | Out | DAC – Channel A – clock |
| SCLK_DAC_B | Out | DAC – Channel B – clock |
| DAC_A [13:0] | Out | DAC – Channel A – data |
| DAC_B [13:0] | Out | DAC – Channel B – data |

## Hardware Interface Layer

All of the signals listed above are connected between the 'User_Ap' interface and the pins of the FPGA via the 'Hardware Interface Layer'. The Hardware Interface Layer includes logic that correctly interfaces many different functional parts of the FPGA, from HERON-FIFO interfaces, to clock inputs and outputs, to digital I/O and serial I/O.

For a complete description of the Hardware Interface Layer (HIL), please read the document 'Using the Hardware Interface Layer in your FPGA Design'.

Interfaces that are specific to this module are :-

### ADC INTERFACE

The global or individual SRC_SCLK signals should be driven with the frequency of operation required for each ADC. The global or individual SCLK signals provide the correct phase of the sample clock to be able to interface to the ADC interface.

Some of the settings discussed earlier in the Config module affect whether the clocks used by the ADC are common or separate. It is important to use the correct settings when using this interface.

The Low-to-High transition on the SCLK output indicates a new sample value is available. The frequency is governed by what you drive into SRC_SCLK. The maximum frequency of this clock is 125MHz.

The 12-bit output bus ADC_n[11:0] is the data sampled from the A/D converter. The interface will produce sample data on each clock cycle.

The output OTR_n is the Out-of-Range signal from the A/D converter. For each clock cycle, the value on the OTR_n output is directly associated with the sample data on the ADC_n output bus.

**DAC INTERFACE**

This interface is used to send data to the D/A converters.

The SCLK_DAC_n input must be connected to a clock signal at the frequency at which the D/A converter is to run.

The 14-bit input bus DAC_n[13:0] is the data to be output to the D/A converter. The data is registered by the component HE_DAC_n on the rising edge of the SCLK_n clock input.

The maximum frequency of the clock input is 125MHz.

## Important!

There are many signals that are connected between the FPGA on the HERON-IO2 and the HERON module connectors. Most of these signals will only be used by advanced users of the HERON-IO2. The FPGA pinning of these signals is defined in the UCF file, and the top.vhd has these signals commented out. Users that want to use these signals will need to uncomment them in the top.vhd and add the correct ports to the user_ap.vhd.

The FPGA sets any I/O pins of the device that are not listed in the design to have a 50-150K pull down. Most of the HERON module signals are pulled to their inactive state by 10K resistors so this 50K will have no effect. However the UDPRES signal does not, and setting this signal low will cause your whole board to be reset. Thus it is important that the UDPRES pin is driven high by the FPGA if it is not being used.

It is also advised to do the same with the LED pins to prevent them becoming illuminated erroneously.

## Other Examples

There are some other examples (source and .rbt files) provided for the HERON-IO2 on the HUNT ENGINEERING CD. Follow “Getting Started” and then “Starting with FPGA”. Choose “General FPGA Examples” and click on the directory for the io2v*. The examples are in the directories example2, example3 etc.

These examples have pdf documents that describe their function.

These examples could be useful to users who do not want to use the FPGA on the HERON-IO2 and simply want to use it as a fast A/D module for a DSP system. Refer to the pdf files to see the functionality provided by each “standard” bitstream.

Please note that as with the examples any “user” work should be done in the user_ap section i.e. do not put your own logic into the hardware interface layer, but simply include them into your own design. This enables your design to be protected from hardware specific details like pinout, and also allows you to benefit from any new versions that HUNT Engineering might make available without having to re-work your part of the design.

## How to Make a New Design

For any new design that you make, it is important that you start from the examples provided on the HUNT ENGINEERING CD.

When making a new design for the HERON-IO2 by starting from one of the examples you will already have a project that is correctly set up to use the supplied Hardware Interface Layer. The project will already include the correct settings and user constraints.

In fact, in all situations you should start development from one of the examples on the HUNT ENGINEERING CD, even if you intend to develop the FPGA in a way that is completely different to any of the examples.

In the case where you are to make a new design that does not match a standard example, you should start development from Example1 and add your own logic in place of the existing Example1 VHDL. By doing this, you will automatically inherit the proper ISE settings, user constraints and project structure.

When your are creating a new design from one of the standard CD examples you will need to be sure that the version of ISE design tools you are using matches the version of ISE for which the example projects were created. If you are using a different version of ISE then you must work through the HUNT ENGINEERING application note 'Using Different Versions of ISE'.

In developing new VHDL, there are proper training courses that exist to help you quickly acquire the required skills and techniques. Search locally for suitable training on these subjects. You may also consider sub-contracting part or whole of the new FPGA design.

### Inserting your own Logic

When making a new design, you will create and insert your own logic inside the USER_AP module.

From here you can interface to the HERON FIFOs, the HSB, the analogue I/O and the general purpose digital I/O.

When these interfaces are simple, you may code the proper logic directly in the USER_AP module.
For more complex interfaces, you may code separate entities in separate source files, and instantiate these entities within USER_AP, as was done in the Examples.

Important: the first thing to edit in the user_ap.vhd file is the package section where generic parameters are set to match your configuration and your design.

Important : The HE_USER interface cannot be left entirely unconnected. If you have a design that does not use the features of this interface, you must be certain to connect the following. The Clock of the HE_USER must be running. The inputs MSG_SEND, MSG_SEND_ID, MSG_LAST_BYTE and MSG_CS must be connected to 0. The MSG_READY must be connected to '1'.

### Top-level fine tuning (using other special IO pins)

The top level defines all of the I/O pins from the FPGA. Some of them are not used in the examples, but have buffers instantiated in the top.vhd. Some of those pins can have alternative signal formats that require a different Xilinx primitive to be instantiated for the buffers.

Refer to the hardware details section of this manual to learn which pins are suitable for which use.

If the buffers that are already instantiated in top.vhd (usually LVTTL) are suitable for your needs then there is no need to modify top.vhd, you can simply use the signals that are connected as ports to the user_ap file.

If you need different buffer types then it is necessary to edit top.vhd.

The method to do that is:

Make a copy of the original TOP.vhd file (from /Common) and work on this copy.

Each I/O pin has a buffer type instantiated in top.vhd.

Edit the instantiation to use the proper Xilinx Buffer primitive.
You may sometimes have to insert attributes in the UCF file to qualify the IO.

Modify the User_Ap entity to make these signals visible.

Add the signals in the User_Ap instantiation port map .

### User Timing Constraints

As with all FPGA designs it is necessary to apply some timing constraints to the design to ensure that the tools generate a design that will operate at the frequency that you require. These will be defined in the .ucf file.

The .ucf file provided as a template has some timing constraints already, but when you make changes to the design you may find that you introduce more clock nets that need to be added to the ucf file.

For more details on Timing Constraints please refer to the Xilinx tools documentation.

## Hints for FPGA designs

Having said that we cannot support you in making your FPGA design, we always try to make your development easy to get started, so this section outlines some things that you need to think about.

The FPGAs are basically synchronous devices, that is they register data as it passes through the device – making a processing pipeline. It is possible to apply asynchronous logic to signals but the FPGA concept assumes that logic is between registers in the pipeline.

This pipeline gives rise to two things that you need to worry and care about. One is the maximum clock frequency that that pipeline can operate at, and the other is the number of pipeline stages in the design.

As with any component in your FPGA design, components from the HERON-IO2 hardware interface layer operate synchronously. That is, any control or data signal that you connect to the interface must be generated from logic that uses the same clock signal that is connected to the library component. Similarly, logic that is connected to outputs of the library component will need to be clocked by the same clock signal.

For a conventional circuit design, you would normally need to consider the signal delays from the output of one synchronous element to the input of the next element. By adding up a ‘clock to output’ delay from the output of the first element, adding routing delays and the ‘setup to clock’ delay for the input of the second element you would have a timing figure to match against the clock period. If the calculated figure is found to be too large, the circuit must be slowed down, or logic must be simplified to reduce the calculated value to one that fits the requirements.

When creating a design using the Xilinx development tools however, you only need to add a timing specification to the clock net that is used to clock both elements. This specification, which may be supplied in units of time or units of frequency will be automatically used by the tool to check that the circuit will run at the specified speed.

This leaves you free to focus on the functionality of the signals, while the Xilinx implementation tools work on achieving your specified time constraints. If at the end of your implementation the tools tell you that your timing constraints have been achieved, then the combination of all setup, hold and routing delays are such that your design will operate at the frequency you defined.

What this means for your design is that when you connect to the Hardware Interface Layer you need to consider whether signals are set high or low correctly on each clock edge (note, all library components are positive/rising edge clocked). What you do not need to worry about is the timing issues of each signal beyond having applied a time constraint on to the clock net that is applied to those components and the connected logic.

## Use of Clocks

Because of the assumption that the design is a pipeline, the development tools will allow you to enter a specification for the clocks used in the design. This allows you to specify the frequency that you need the resulting design to operate at.

Simple designs will use only one clock, and all parts of the design will use that clock. It is usual for every “part” of the design to use the rising edge of this clock. This makes it very simple for the development tools to determine the maximum possible frequency that design could be used at. Adding too much combinatorial logic between pipeline stages will reduce the maximum possible clock rate. This gives rise to a hint – If your design will not run fast enough, add some more pipelining to areas where lots of combinatorial logic is used.

Typically development tools will give a report stating the maximum clock rate that can be used in a particular design, and will probably raise errors if that is slower than the specification that you have provided for the clock used in the design.

More complicated designs would use several clock nets, which may be related in frequency or phase, or may be completely independent. In such a design you must be careful when outputs from logic using one clock are passed to logic using a different clock. It is often useful to add a FIFO, which allows input and output clocks to be completely independent.

## Possible Sources of Clocks

As you can see from the section above, an FPGA design may require one or more clock frequencies to achieve the job it needs to do. How *you* implement *your* design governs the number and frequencies of the clocks you need. The design of the module has been made to give you as much flexibility as possible, but ultimately it is up to you which will be used.

On the HERON-IO2 there are four possible Crystal oscillator modules, an AC coupled low level clock input and four external clocks possible from the digital I/O connector. Any of these can be used as clocks for in the FPGA design, as can clocks provided on any of the other I/Os. One technique for example could be to use one of the UMI pins as a clock input, which can be driven by the timer of a DSP module, or possibly another FPGA module driving a clock onto that UMI. This type of use though is system specific, and we cannot supply a generic example for that. The examples that we provide for the module assume a clock is fitted to the UserOSC3 position, and use that as the only clock in the design.

### Flow Control

Because the processing speed of the FPGA will almost never be the same as every other component in your system it will be necessary to use some flow control in your design. The most general way to implement this is to use Clock enables to enable the processing only when it is possible for data to flow through the "system". Otherwise some type of data storage (like FIFOs) must be implemented to ensure that data is never lost or generated erroneously.

When data is read from the HERON Input FIFOs there are FIFO flags to indicate when there is data to be read. Reading from the FIFO when the flags indicate that there is no data to read will result in false data being fed through your system. Thus your design must either a) only assert the read signal when the Empty Flag (EF) is not asserted, or b) use the EF as a clock enable for the logic in the design, thus preventing the invalid data caused by reading an empty FIFO, from being propagated through the design. The actual method used will depend on the needs of the design.

When data is written to the HERON Output FIFOs there are flags to indicate when it is possible to write new data. Writing to the FIFO when the flags indicate there is no room will result in data being lost from your system. Thus your design must not assert the Write enable signal when the Full Flag (FF) is asserted.

### Pipeline Length or "latency"

The latency of your design will be determined by the length of the pipeline used in your FPGA design. The simplest way to determine this is to "count" the Flip-Flops in your data path, but whichever tools you are using might provide a more elegant way. For example the "Core Generator" will state the pipeline steps used with each core, and will even let you specify a maximum in some cases.

## Digital I/O from the FPGA

In addition to the FIFO and HSB interfaces of the HERON-IO2, there are some General purpose Digital I/O connectors and some serial interfaces. The use of these interfaces will be specific to a particular design, so they have not been included in the examples supplied. The pins to use are defined in the top.vhd, and the locations are already defined in the .ucf files. The choice of buffer type and the time specs of those interfaces must be taken care of by the designer.

## DSP with your FPGA

The FPGA can be used to perform powerful Digital Signal Processing. It is beyond the scope of this manual and indeed not part of the normal business of HUNT ENGINEERING to teach you how to do this. There is however a simple way to build Signal Processing systems for the Xilinx FPGAs.

Xilinx supply as part of their toolset something called a "Core Generator". This provides a simple way of generating filters, FFTs and other "standard" signal processing elements, using a simple graphical interface. It results in a "block" that can be included in your design and connected like any other component. Typically it will have a clock input that will be subject to the pipeline speed constraints, and clock enables to allow flow control.

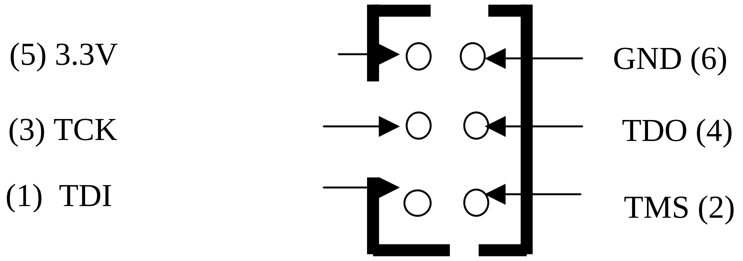
Using the Core Generator you can quickly build up signal processing systems, but for more complex systems you should take a course on Signal Processing theory, and perhaps attend a Xilinx course on DSP using FPGAs.

## Chipscope

It is possible to use 'Chipscope' software, from Xilinx, to debug your FPGA. 'Chipscope' links to the IO2V through the JTAG PROM Programming connector as this is also linked to the JTAG pins of the FPGA.

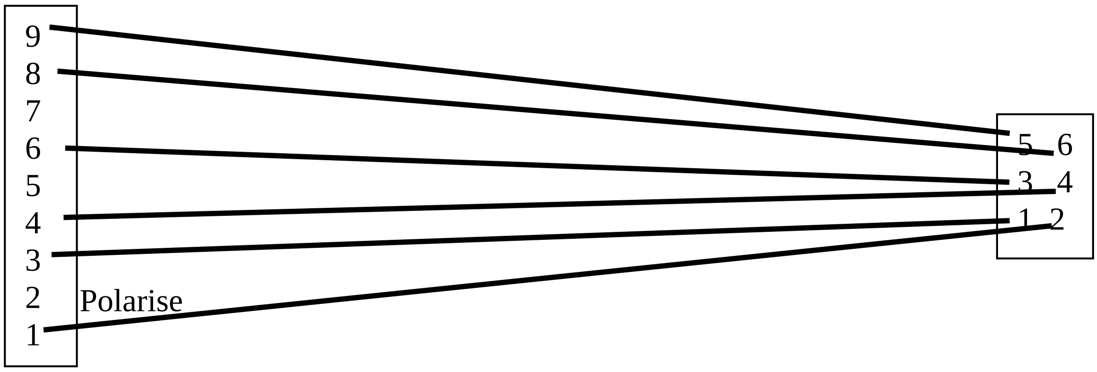
The JTAG connector fitted to the IO2V is a Hirose 2mm 3x2 connector of part number DF11-6DP-DSA(01). The mating half that is required for cabling is also a Hirose part, the housing is DF11-6DS-2C and crimp part number DF11-2428CSA.

The pinout is:-



Top view of connector.

The cable supplied with IO2V modules is as follows:-



Which can be fitted to a Xilinx JTAG cable (such as Xilinx parallel cable 3 or 4) and used to connect to the Virtex II on the module.

Further information about 'Chipscope' can be found at the Xilinx web site.

The software for use with your FPGA will consist of several parts-

## FPGA Development tool

The application contents of the FPGA will be generated using FPGA development tools. Xilinx ISE is the recommended tool, along with ModelSim if you require simulation.

It is possible to use alternative FPGA synthesis tools such as Leonardo Spectrum, or Synplicity, but ultimately the Place and Route stage will be performed by the same tool. Users of ISE have the Place and Route tool included, but users of the other tools require the Xilinx Alliance tool.

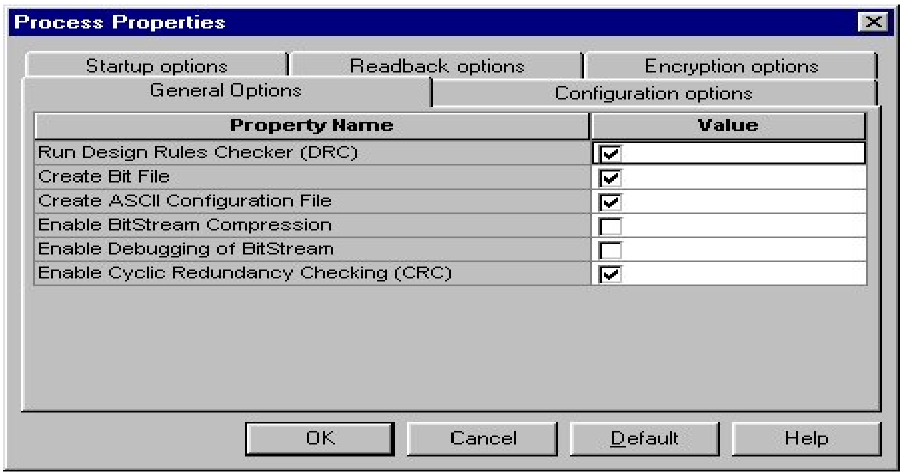
The FPGA design can be entered using VHDL.

## Design Files for the FPGA

The FPGA design can be downloaded onto the HERONIO2V2 in three ways. Via the Configuration serial bus which requires a *.rbt file, via the Flash PROM on the JTAG chain which requires a *.mcs file, and thirdly directly programming the FPGA via the JTAG chain which requires a *.bit file.

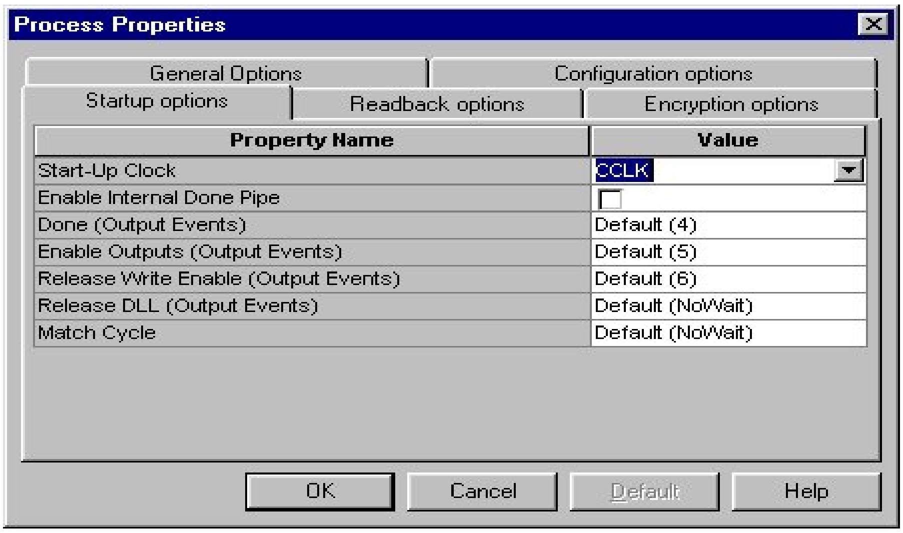
## Files for HERON Utility (*.RBT)

The sections in this document titled “Creating a Project” and “Inserting your own logic” lead up to the generation of a *.rbt file which can be downloaded via the Configuration serial bus using the HERON Utility. Before generating the *.rbt file right click on “generate Program File” in the “processes for Current Source” window in ISE , select “Properties” on the menu and then select “General Options”. Check that “create ASCII Configuration File “ has been selected.



| Property Name | Value |
|---|---|
| Run Design Rules Checker (DRC) | ☑ |
| Create Bit File | ☑ |
| Create ASCII Configuration File | ☑ |
| Enable BitStream Compression | ☐ |
| Enable Debugging of BitStream | ☐ |
| Enable Cyclic Redundancy Checking (CRC) | ☑ |

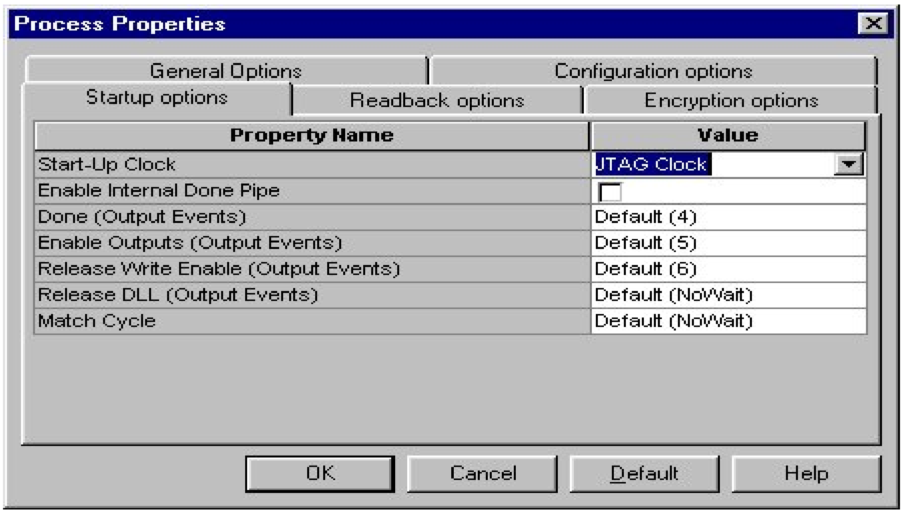
Also from “Process Properties” select the “Start-up Options” and check that CCLK has been selected for the “Start-Up Clock”. This is the default used in the example projects provided.



| Property Name | Value |
|---|---|
| Start-Up Clock | CCLK |
| Enable Internal Done Pipe | ☐ |
| Done (Output Events) | Default (4) |
| Enable Outputs (Output Events) | Default (5) |
| Release Write Enable (Output Events) | Default (6) |
| Release DLL (Output Events) | Default (NoWait) |
| Match Cycle | Default (NoWait) |

### Files for direct JTAG programming (*.bit)

The FPGA can be programmed directly through the JTAG connector on the HERONIO2V2 using Xilinx "iMPACT" software together with a *.bit file. The only difference between the option settings for the *.rbt file is in the "Start-Up clock" which should now be set to JTAG Clock.



| Property Name | Value |
|---|---|
| Start-Up Clock | JTAG Clock |
| Enable Internal Done Pipe | □ |
| Done (Output Events) | Default (4) |
| Enable Outputs (Output Events) | Default (5) |
| Release Write Enable (Output Events) | Default (6) |
| Release DLL (Output Events) | Default (NoWait) |
| Match Cycle | Default (NoWait) |

### Files for PROMs (*.MCS)

The Flash PROMs on the FPGA3 can be programmed, and reprogrammed via the JTAG chain using '*.mcs' files. These files are generated by the Xilinx "PROM File Formatter" after the '*.rbt' file has been generated.

Please read the document "Using iMPACT with FPGA modules" for a detailed description of how to create the correct '.mcs' file for your design.

## Downloading Files via JTAG

Once the *.mcs and/or the *.bit files have been generated they can be downloaded to the PROMs or directly to the FPGA using the Xilinx "iMPACT" software.

Please read the document "Using iMPACT with FPGA modules" for a detailed description of how to do that.

## HERON_FPGA Configuration Tool

HUNT ENGINEERING provides a tool to allow you to load the FPGAs in your system with .rbt files that you create, or copy from the CD.

For details about using this tool refer to the "Started your FPGA development" tutorial on the HUNT ENGINEERING CD.

The windows tool actually calls a program with command line parameters set according to your choices.

The program is HRN_FPGA.exe which will have been installed on your DSP machine in the directory %HEAPI_DIR%\utils.

For help using that program directly type hrm_fpga –h in a DOS box.

## HUNT ENGINEERING HOST-API

The HOST-API provides a consistent software interface to all HERON Module carriers, from a number of operating systems.

While the FPGA development tools can only be run under Windows on a PC (some can be obtained in Unix versions for a workstation). It is possible to deploy your system from a number of different Host operating systems. In these cases the HOST-API and the FPGA loader tool can allow you to use your system, even if you cannot make your FPGA development there.

Refer to the tutorials, documentation and examples for the HOST-API on the HUNT ENGINEERING CD.

## HUNT ENGINEERING HERON-API

If you have C6000 DSPs in your system, you can use HERON-API to communicate with the FPGA modules via the HERON-FIFOs. Refer to the tutorials, documentation and examples for the HERON-API on the HUNT ENGINEERING CD.

## HERON Module Type

The HERON-IO2 module implements all four of the HERON connectors, which means it is a 32 bit module that can access all twelve of the possible HERON FIFOs.

For a complete description of the HERON interfaces, signal definitions and connector types and pin outs, refer to the separate HERON specification document. This can be found on the HUNT ENGINEERING CD, accessed through the documentation viewer, or from the HUNT ENGINEERING web site at http://www.hunteng.co.uk.

The HERON-IO2 does not have a processor so does not assert the “Module has processor” pin as defined in the HERON specification.

The HERON-IO2 does not support JTAG so does not assert the “Module has JTAG” pin as defined in the HERON specification.

The HERON-IO2 has a serial bus so asserts the “Module has serial bus” pin as defined in the HERON specification.

The HERON-IO2 has a 32 bit interface so asserts the “32/16” pin low.

## Hardware Reset

Before the HERON-IO2 can be used, it must be reset. This reset initialises the Serial bus circuitry into a state where it can be used. Depending on the way that the user FPGA was last configured, it may also reset some functions in the user FPGA.

This reset DOES NOT cause the user FPGA to require re-configuration.

This signal is driven by the HERON module Carrier and must NOT be left unconnected, as this will cause the HERON-IO2 to behave erratically. It must also NEVER be driven by the FPGA on the HERON-IO2.

## Software Reset (via Serial bus)

The Serial configuration bus has a reset command that is executed at the beginning of a bit stream download. This must never be confused with the system hardware reset provided on the HERON pin – it is not the same thing. The Serial bus reset simply resets the internal configuration of the FPGA but will NOT perform a hardware reset.

It cannot affect the HERON carrier board FIFOs, or any other module in the system.

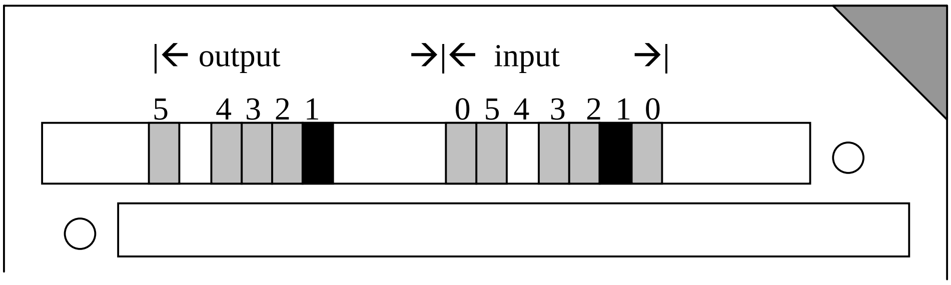
## Config

There is a system wide Config signal that is open collector and hence requires a pull up to be provided by the motherboard. The HERON-IO2 can drive this signal active (low) or inactive if required, or can use it as an input to disable data transfer during a DSP booting phase.

If the User FPGA program does nothing with this signal the signal will be pulled high by the carrier board.

## Default Routing Jumpers

The default routing jumpers are provided by HERON modules. These are the longer pins on the topmost HERON connector of the module. These pins protrude above the HERON module when it is fitted to the module carrier, to which jumper links can be fitted.



These jumpers can be used to select the default routing of FIFO 0 on the Carrier card.

The exact use of these jumpers is defined by the HERON module carrier, so you should reference the user manual for the Carrier card you are using, but the numbering of these “default routing” jumpers is defined in the HERON spec, and shown above.

e.g. the jumpers as fitted in the diagram show that the default routing for both the input FIFO #0 and the output FIFO #0 is selected as 1.

HERON processor modules accept their boot stream from FIFO 0, which is why these jumpers are provided for FIFO #0 only.

The HERON-IO2 does not need to boot from FIFO 0, so the Default Input FIFO 0 can be set or not set as required by the user.

## Physical Dimensions of the Module

A size 1 HERON module is 4.0 inches by 2.5 inches overall.

The 5mm limit on component height under the module is not violated by the HERON-IO2.

The maximum height of the HERON-IO2 above the module is 6.5mm including mating connectors and cables.

This means that the assembly of a HERON module carrier and the HERON-IO2 is less than the 20mm single slot spacing of PCI, cPCI and VME.

## Power Requirements of the HERON-IO2

The HERON-IO2 only uses power from the 5V and 12V HERON supplies. The 3.3V and 1.5/2.5V required by the FPGA are generated on board from the +5V. The analogue +5V and 3.3V are generated from the +12v.

The maximum rating of the HERON-IO2 is determined by the number of gates and the actual FPGA program loaded.

The other logic on the HERON-IO2 has a maximum of 150mA at 5V, and the analogue section will take a maximum of 500mA at 12V.

The Switch mode circuits on the HERON-IO2 for the FPGA can each supply 1.5A, i.e. 1.5A at 1.5/2.5V and 1.5A at 3.3V. These circuits are at least 80% efficient.

## FPGA Power consumption/Dissipation

The power consumption of an FPGA is governed by the number of edge transitions per second. This means it depends on not only on the configuration loaded into it and the clock frequency but also on the data being processed.

The flexibility of a Xilinx FPGA means that determining the possible power consumption is not a trivial task, an estimate can be obtained by using the Xilinx 'XPower' software package. It is still difficult to get an accurate measure because you need to describe the real data values and timings to be able to estimate correctly.

The FPGA package on the HERON-IO2V2 is an FG456 which can dissipate 2.4 watts maximum with an ambient temperature of 50 deg C.

The HERON-IO2V2 power supplies for the FPGA are capable of delivering:-

| | | | |
|---|---|---|---|
| 1.5Volt | @ | 1.5Amps | 2.25Watts |
| 3.3Volts | @ | 1.5Amps | 4.95Watts |
| | | Total | 7.2Watts |

This is well above the bare package maximum power dissipation. This means that the first limit on power consumption of the HERON-IO2V2 is determined by the FPGA package.

It is always a good idea to have some airflow past the package, and normally a Fan fitted to the Case of the PC is sufficient to provide this.

The dissipation limit of the package can be increased by fitting a heatsink and possibly a fan. Depending on the performance of this heatsink your FPGA design could then reach the limit of the power supply circuits on the module. In the unlikely event that this second limit is reached it will be necessary to modify the module to use external power supplies for your system.

## FIFOs

The HERON FIFO connections are shown in the table of FPGA pinout. The timing of the signals can be found in the HERON module specification which is on the HUNT ENGINEERING website and CD.

The design implemented in the FPGA MUST drive a constant clock onto the FIFO clock pins. The clock driven by the FPGA on “O/P FIFO clock” and “I/P FIFO clock” is buffered with an LVT245 buffer that has enough current drive for the carrier board clock signal. The actual phase of the clock on the FIFO will be the same as the inputs on GCLK2 and3, so DLLs can be used to use that same clock to drive logic in the FPGA.

There may be a minimum and maximum frequency imposed by the module carrier that the module is fitted to.

However it is not necessary to look up any of this information as we supply a “library” component that can be placed in your design and will take care of the FIFO accessing for you.

## User FPGA Clocking

As has been said elsewhere in this manual, the clocking of the FPGA can be a complex issue. The FPGA does not have such a thing as a clock pin, but rather can use an I/O pin as a clock, for almost any part of the FPGA design.

Your design will be simpler if a single clock is used, or even if there are several clocks used, but they are derived from each other. However the FPGA must drive the input and output FIFO clocks, the ADC clocks, and the DAC clocks. In each case the logic that interfaces to each must be clocked from the same clock as the interface.

Different carrier boards have different requirements for the FIFO clocks, and different applications will have different needs for the sampling rates of the ADC and DAC interfaces. If they can all be the same then this is the simplest case, and a single clock input to the FPGA is needed. When the HERON-IO2 is shipped there is a 100Mhz oscillator fitted to User OSC3.

If the rates of those clocks need to be different but can be derived from each other then the design can still be kept quite simple, but sometimes it will be necessary to have several completely independent clock presented to the FPGA.

To allow a large amount of flexibility, the HERON-IO2 offers up to 4 Oscillator modules, an AC coupled low level clock input and also a digital I/O connector where clocks can be input to the module from a cable. There are also other possibilities like the Digital I/O signals or the UMI pins of the HERON module.

Clocks inputs can be used directly in your FPGA design, but Xilinx provide Delay Locked Loops (DLL) in the FPGA design. These can be used for a number of purposes such as clock multipliers, or to align the phase of an internal clock with that of a clock signal on an I/O pin of the device. This second way is used by the Hardware Interface layer to guarantee data access times on some of the interfaces.

GCLK3S is driven from the buffered Output FIFO clock, allowing a DLL to be used to synchronise the internal logic to that clock. This is used by the Hardware Interface Layer to manage the FIFO clocking.

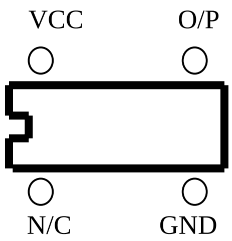
GCLK6P is driven from the buffered Input FIFO clock, allowing a DLL to be used to synchronise the internal logic to that clock. This is used by the Hardware Interface Layer to manage the FIFO clocking.

The “AC Clock” input is connected to GCLK4S and GCLK5P as a differential pair.

## User oscillators

The OSC0 socket on the HERON-IO2 accepts a plastic bodied TTL oscillator such as the SG531 type from Seiko Epson. The package is a 0.3" 8 pin DIL type body but with only 4 pins

The socket has four pins as shown



These devices are available in TTL and 3.3V versions. Either can be used as the output is buffered by a 5V tolerant buffer before it is connected to the FPGA pin. The socket provides +5v supply.

OSC1 is a surface mount location where an oscillator can be fitted at build time. This site is also powered from 5V and is buffered with a 5V tolerant buffer.

OSC2 and 3 are also surface mount locations that can be populated at build time. They are powered from 3.3V and the outputs are NOT buffered before connection to the FPGA. OSC3 is fitted with a commercial grade (_+100ppm) 100Mhz oscillator at the factory.

The above part number is just an example of the oscillator type that can be fitted, and the exact specification of the oscillator should be chosen carefully for the application it will be used for. For example the tolerance, jitter and temperature dependence might be important considerations for some applications.

## AC CLOCK Connector type

There is a microminiature 50 Ohm coaxial connectors provided for direct input of an A/D sample clock. It can be connected to a low level sinusoidal input, or an LVTTL input.

The Connector is manufactured by Radiall, and are their type "MMT". The board connector is Radiall Part number R210408012.

We purchase a cable assembly that has the mating connector and 200mm of cable type RG174. This assembly has Radiall part number R284008001. Usually if you have explained at the time of ordering how you will be using your HERON-IO2 module there will be cabling supplied that suits your needs.

## ESD protection

All of the Clock I/Os are protected against Electro Static Discharge and over voltage The devices used are Harris SP723 parts.

This protects the inputs to IEC1004-2 level 4, and provides over voltage limiting to the range 0 to +5V.

## Analog I/O

The HERON-IO2 connects the FPGA to two 12 bit A/D chips that can sample at rates between 1 and 125Mhz. This allows the User FPGA program to sample the A/D data, process it, store it or pass it on to another HERON module as the user's system requires.

### Analog input connector type

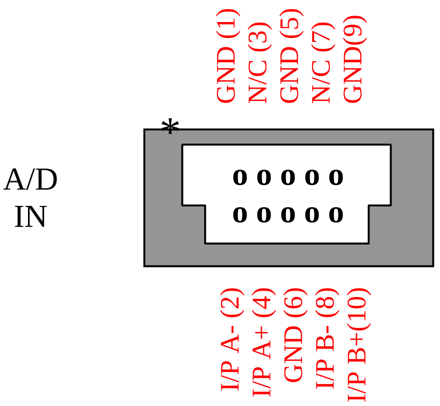
The analog inputs to the HERON-IO2 are differential, and are both connected via the "A/D IN" connector. It is arranged as 5 pins in each of 2 rows. It is supplied by Hirose and its part number is DF13-10DP-1.25V(50) . This connector has polarisation against incorrect insertion and mechanical retention of the mating half.

The mating connector is also supplied by Hirose and has part number DF13-10DS-1.25C which requires crimp contacts part number DF13-2630SCFA. These crimps are only available from Hirose in large quantities and require special tooling. Usually if you have explained at the time of ordering how you will be using your HERON-IO2 module there will be cabling supplied that suits your needs.

If your requirements change then HUNT ENGINEERING will be able to supply assemblies or component parts to meet your needs but a charge will apply.

### Analog input connector pinout

The input connector is labelled 'A/D IN'.



The I/P n+ and I/P n- signals form the differential input pair. The GND connection is not normally connected but is provided to allow the connection of system grounds in the D/C coupled case.

### A/D data

The A/Ds used on the HERON-IO2 are the AD9433-125 part from Analog devices. This has a maximum clock frequency of 125MHz, and the data sheet for the part states that the dynamic performance will degrade if the sample rate is below 10MHz.

Each of the A/Ds are connected to the FPGA directly, and provide 12 bit data in 2's complement format. There is also an over-range bit provided by the parts that are also connected to the FPGA.

| Code | Ain+ Ain- difference | Digital output | Over-range bit |
|---|---|---|---|
| +2047 | > 1V | 0111 1111 1111 | 1 |
| +2047 | 1V | 0111 1111 1111 | 0 |
| 0 | 0V | 0000 0000 0000 | 0 |
| -1 | –0.00049V | 1111 1111 1111 | 0 |
| -2048 | -1V | 1000 0000 0000 | 0 |
| -2048 | < -1V | 1000 0000 0000 | 1 |

## A/D missing codes

Although the AD9433 data sheet guarantees no missing codes over the full temperature range, it has been observed that when the inputs are close to or over the full range of the A/D that there are some codes missing near the limits of the range. For example there may be no codes above 0x7e0. It has been proven that this is the converter component and not the analogue or digital connections to it.

HUNT ENGINEERING test each module for no missing codes between 0x810 and 0x7e0 at room temperature.

## A/D clocking

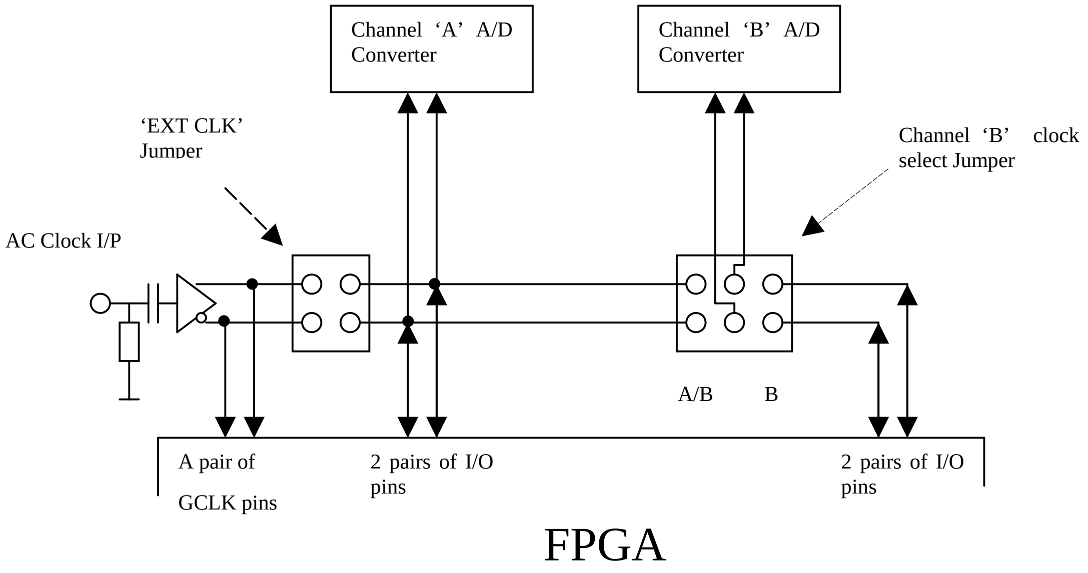
The A/Ds can be clocked by the FPGA to allow control of the sampling rate, which can be derived from any of the User Oscillator sockets, external clock inputs or even HERON UMI pins.

The sample clocks for the A/D converters are differential LVPECL with the differential input to the converters AC coupled to meet the common-mode DC levels required by the AD9432 converter.

Each A/D has a separate clock so that they can be driven with different frequencies or phases according to the user FPGA configuration, but there is a jumper that allows the same clock to be selected for both channels to remove any phase error that might occur between two separate clocks.

There is also the option of clocking the A/D's from an external source, the "AC Clock" input. This is an AC coupled 50 Ohm input designed to accept sinusoidal signals of level approximately 0dBm, and drives a LVPECL differential receiver with sub nanosecond propagation delay. The resulting LVPECL signal is connected directly to a pair of GCLK pins on the FPGA, and also to a pair of two pin jumpers, one each for the +ve and –ve differential LVPECL signal which can be used to link the clock signal to the channel A A/D converter.

There is a jumper option to clock the channel B converter with the same clock as channel A or from a separate clock generated from the FPGA.

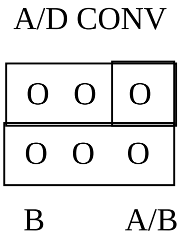


When the AC Clock input is used to drive the sample clocks then the FPGA must be configured such that none of the I/O pins on the FPGA are outputs driving against the signal from the AC Clock input.

The lower frequency limit of the "AC clock" input for clocking the A/D converters is approximately 4 MHz when driving the input with a 0 dBm sine wave. For frequencies below 4Mhz one of the digital clock inputs should be used.

## A/D clock jumper

There is a jumper near to the FPGA as follows:-



The centre pins are connected to the encode pins of the A/D part B. If a jumper link is fitted between the pin labelled B and the centre pin, the encode signal will be driven by the encode B output from the FPGA. If however a jumper link is fitted between the pin labelled A/B and the centre pin, the encode B signal will be driven by the encode A output from the FPGA, making both A/Ds use exactly the same phase of clock.

## A/D pipeline delay

The AD 9433 has a twelve clock pipeline delay internally. That is the digital value presented on its pins after the rising edge of the encode signal, actually represents the analogue value that was on the input of the A/D twelve encode cycles previously.

The A/D pins are connected directly to the FPGA pins with no buffer delay.

The Hardware Interface Layer registers this data into the FPGA (using an IOB register to guarantee the timing needs are met) introducing another (13th) pipeline delay.

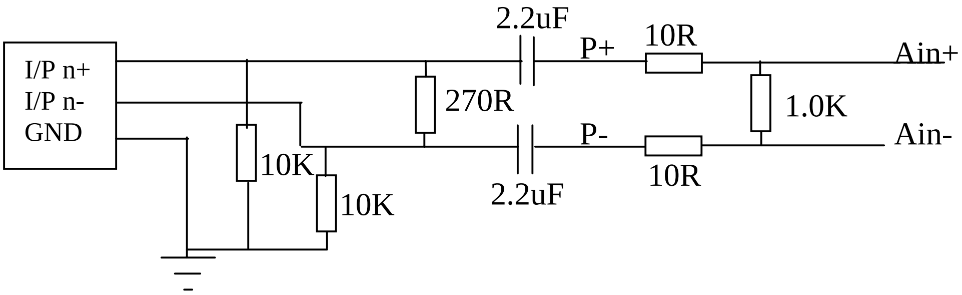
This pipeline delay makes the HERON-IO2 better suited to streaming data applications such as data acquisition or for use in communications systems than in control loop type applications.

## Analogue input options

There are some build options for the analogue inputs of the HERON-IO2.

**Standard A/C coupled**

The standard analogue input configuration is to be A/C coupled, to have a 200R impedance between the differential inputs.



The ground pin of the input connector is connected to the ground plane of the PCB.

With this input configuration the performance should be :-

**Input Bandwidth**

The AC coupling capacitor on the input defines the low frequency cut off, and the converter chip defines the upper cut off, resulting in an analogue bandwidth of between 750Hz and 500MHz.

**SNR**

The AD9433 data sheet shows a worst case typical SNR of 57.7dB's .

Production test of HERON-IO2s measure for a maximum spread of 8 levels of noise with an unconnected input.

More detailed investigation of this noise shows that it typically has a mean square of 0.45 levels, which is equivalent to a SNR of 65.69dB

Adding the usual "inside the box" cable stubs provided with your system changes this very little, to 0.46 levels (mean square) which is a SNR of 65.69dB

Adding a co-axial external cable to this has quite a large effect, changing the mean square to be 4.3 levels, equivalent to a SNR of 55.89dB.

This shows the care that should be taken when making production cables to connect to the A/D inputs.

### Offset

The worst case offset is defined by the A/D component as +- 12 levels.

### Input level

The Converter chip defines the input level as 2V peak to peak (differential).

### Impedance

The input impedance is set to approximately 200R by the value of the resistor between the input pins. (see circuit above). Optionally the impedance can be set at build time to any value between 50R and 1K, by fitting different values for R26. However this may affect the noise and offset performance.

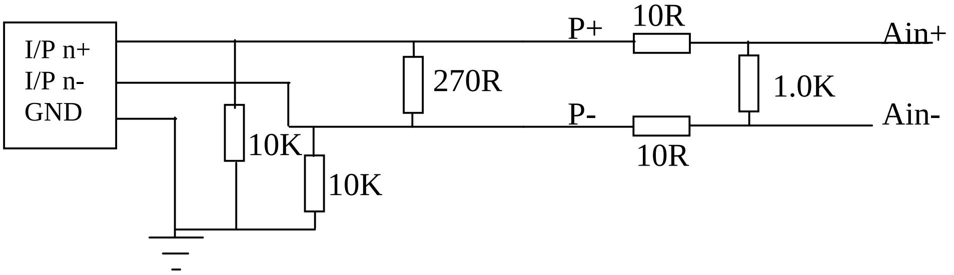
### Protection

The points P+ and P- are protected against overvoltage ( >+5v) and undervoltage (<0v) with respect to the HERON-IO2 ground. They are also protected against static discharge. The voltage rating of two 2.2uF capacitors sets the absolute maximum and minimum input voltages, relative to the HERON-IO2's 0Volts, and these are +15Volts and –10Volts.

### D/C coupled

Optionally a D/C coupled input can be specified at build time.

The standard D/C coupled input configuration is to have a 200R impedance between the differential inputs. This results in a signal bandwidth of 0 to 500Mhz.



The ground pin of the input connector is connected to the ground plane of the PCB.

With this input configuration the performance should be :-

### Input Bandwidth

In this case the A/D component defines the high and low frequency cut offs, resulting in an analogue bandwidth of between 0Hz and 500MHz.

### SNR

The AD9433 data sheet shows a worst case typical SNR of 57.7dB's .

Production test of HERON-IO2s measure for a maximum spread of 8 levels of noise with an unconnected input.

More detailed investigation of this noise shows that it typically has a mean square of 0.45 levels, which is equivalent to a SNR of 65.69dB

Adding the usual "inside the box" cable stubs provided with your system changes this very little, to 0.46 levels (mean square) which is a SNR of 65.69dB

Adding a co-axial external cable to this has quite a large effect, changing the mean square to be 4.3 levels, equivalent to a SNR of 55.89dB.

This shows the care that should be taken when making production cables to connect to the A/D inputs.

### Offset

The worst case offset is defined by the A/D component and the input biasing as +- 16 levels.

### Input level

The Converter chip defines the input level as 2V peak to peak (differential).

In this configuration it is important to keep the signal inputs within the range of the A/D, which is between 0v (GND) and +5V. For the A/D converter **AD9433-125**, this means that when using D/C coupled inputs the signals should have a common mode DC offset of about 4v. i.e each of the inputs should swing from +4.5V to +3.5V **with respect to the HERON-IO2 Gnd signal**.This means that there must be a connection made to the GND pin on the analogue input connector. Care should be taken when doing this to prevent any ground loop problems that could cause noise.

**Impedance**

The input impedance is set to approximately 200R by the value of the resistor between the input pins. (see circuit above). Optionally the impedance can be set at build time to any value between 50R and 1K, by fitting different values for R26. However this may affect the noise and offset performance.

**Protection**

The points P+ and P- are protected against overvoltage ( >+5v) and undervoltage (<0v) with respect to the HERON-IO2 ground. They are also protected against static discharge.

## ESD protection

The analogue inputs are protected against Electro Static Discharge and over voltage The devices used are Harris SP723 parts. The connections are made at the points labelled P+ and P- on the drawings in the sections above.

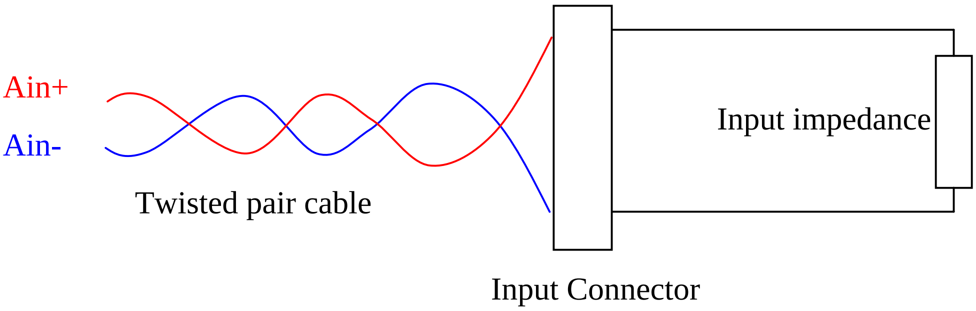
This protects the inputs to IEC1004-2 level 4, and provides over voltage limiting to the range 0 to +5V.

Users should ensure that cabling used in a system enables compliance with EMC directives.

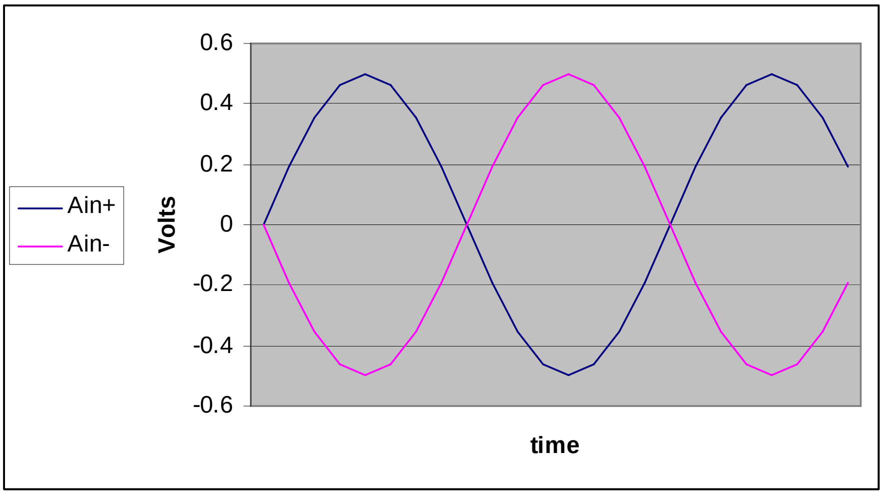
## Differential inputs

Differential inputs are used as these provide a good method of connecting an input with minimum noise. Any noise picked up by the cabling should be the same for both signals ( + and -) so will not affect the signal value that is digitised. The current the flows in each of the wires is also the same which also helps minimise the likelihood of transmitting or picking up noise.

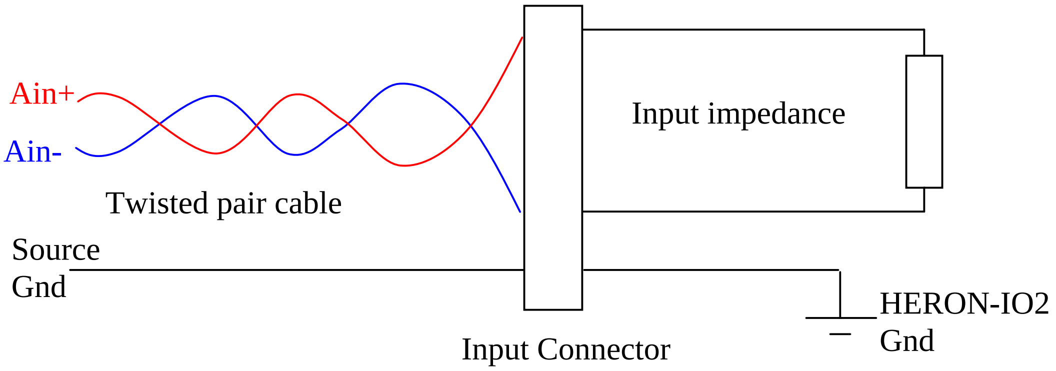
When an A/C coupled differential input is connected the input looks like :-



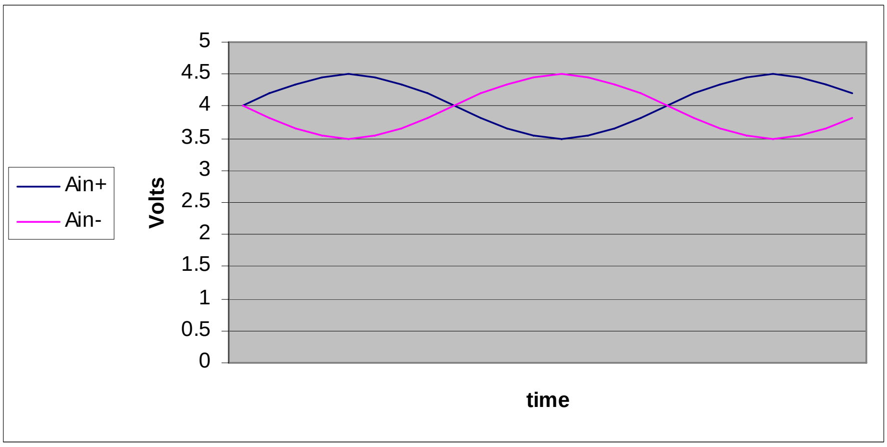
The A/C coupled differential signal input would need to be :-



If the input is single D/C coupled the connection should be :-



Then the differential input must be :-



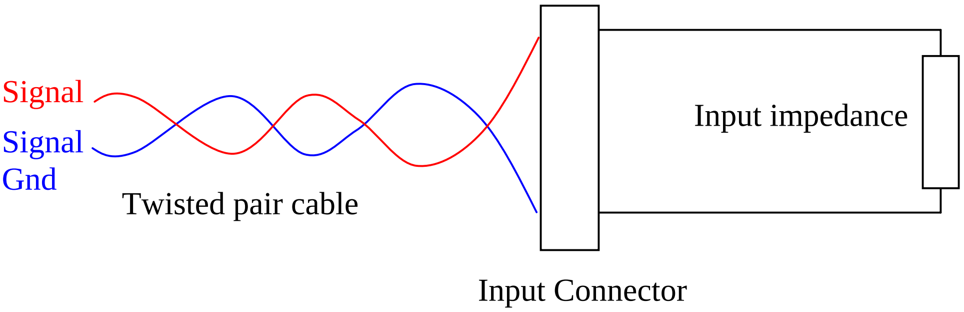
The AD9433-125 A/D converter has a common mode input voltage requirement of 4Volts

.

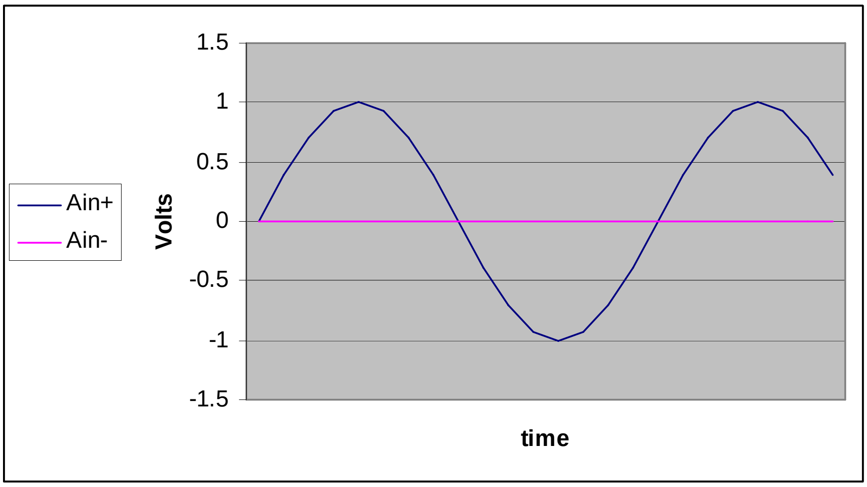
Connecting single ended signal sources

If however the signal source is not differential, a single ended signal source can still be connected to the differential input as follows :-

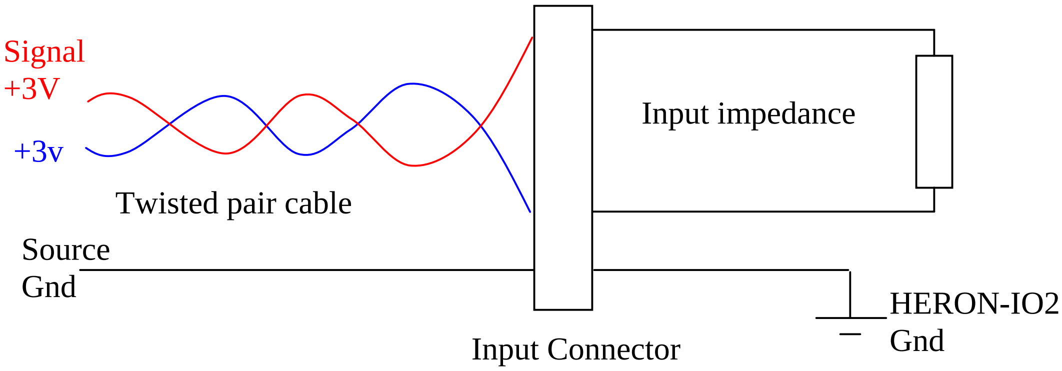
An A/C coupled input



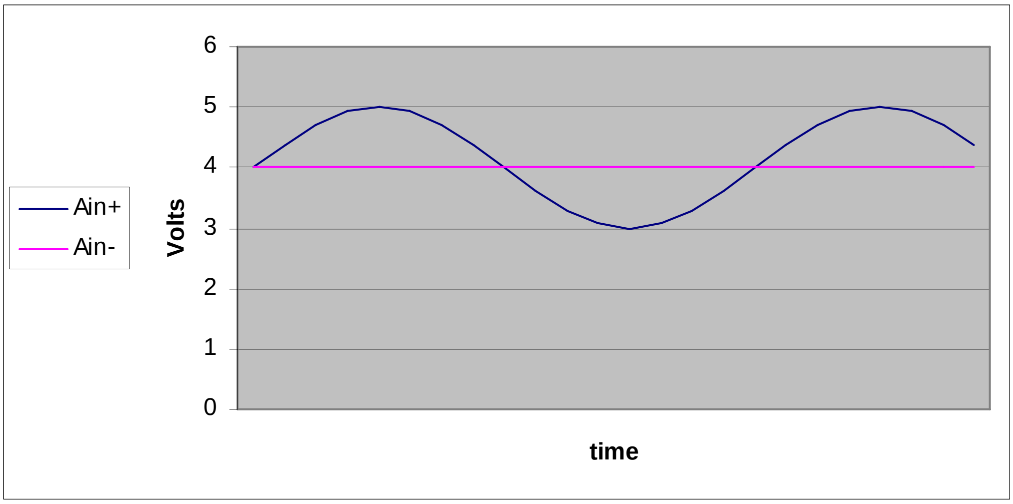
Now the input signals will be :-



For the D/C coupled case it is a little more difficult :-



Where the signals must be :-



The AD9433-125 A/D converter has a common mode input voltage requirement of 4Volts.

This is only possible with a ground connection between the two systems, which should be made by a single system connection to prevent ground loops.

## Cabling precautions

In order to achieve best signal performance, and to achieve compliance with EMC regulations for radiation and susceptibility, the cabling used in a system that uses the HERON-IO2 must be carefully designed.

The use of differential inputs suits the use of twisted pair cabling, which minimises reception and radiation of noise. This cabling has a typical characteristic impedance of around 200R which is why the "standard" input impedance of the HERON-IO2 is chosen as that value.

The twisted pair cabling needs to be individually screened, to help further with noise performance. This screen should NOT be connected to signal ground or the 0V of the power supply, but MUST be connected to the earthed casing of your system.

In accordance with this any cabling that is provided by HUNT ENGINEERING for a HERON-IO2 will use twisted pair cables, with an individual screen for each pair. Normally the cables will be connected to a D type connector on a panel. The screens from the pairs will be connected together, and connected to the panel where the D type connector is mounted AND to a pin of the d type connector. This allows a mating cable to be used that has a screened shell, but rather than rely on the screwlocks (that can become loose and hence high resistance) to make the connection of the screen to the case, the screen can also be connected using the connector pin provided.

## Analog output connector type

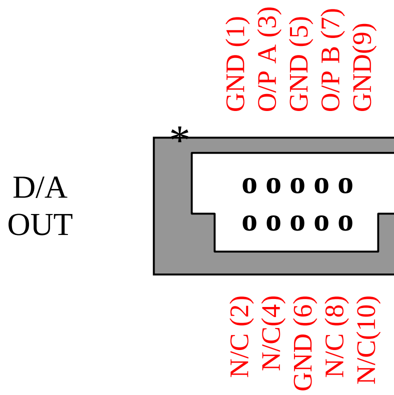
The analog outputs from the HERON-IO2 are single ended, and are both connected via the "D/A OUT" connector. It is arranged as 5 pins in each of 2 rows. It is supplied by Hirose and its part number is DF13-10DP-1.25V(50) . This connector has polarisation against incorrect insertion and mechanical retention of the mating half.

The mating connector is also supplied by Hirose and has part number DF13-10DS-1.25C which requires crimp contacts part number DF13-2630SCFA. These crimps are only available from Hirose in large quantities and require special tooling. Usually if you have explained at the time of ordering how you will be using your HERON-IO2 module there will be cabling supplied that suits your needs.

If your requirements change then HUNT ENGINEERING will be able to supply assemblies or component parts to meet your needs but a charge will apply.

### Analogue output connector pinout

The input connector is labelled 'D/A OUT'.



The O/P n signals are the analogue outputs. They are voltage outputs relative to the GND.

### D/A data

The D/As used on the HERON-IO2 are AD9767 parts from Analog Devices. They have a maximum clock rate of 125Mhz, no minimum clock rate is specified on its data sheet.

Each of the D/As are connected to the FPGA directly, which must provide 14 bit data in offset binary format.

The D/A's analogue output level is related to the input digital value as shown in the table below:-

| INPUT<br>(hex) | ANALOG OUTPUT<br>(Volts) |
|---|---|
| ^h0000 | -1 Volt |
| ^h1fff | 0 Volt |
| ^h3fff | +1 Volt |

However this voltage is on the output of the output buffer, and will be divided across the series resistance in the output of the HERON-IO2 (10R) and the load impedance.

For example a 50R load will have 50/(50+10) x2V across it which is 1.66V peak to peak.

### D/A clocking

The D/As are clocked by the FPGA to allow control of the sampling rate, which can be derived from any of the User Oscillator sockets, external clock inputs or even HERON UMI pins. The clock from the FPGA is connected to the CLK1 and CLK2 inputs of the D/A part directly. There are separate connections from the FPGA to the WRT1 and WRT2 signals of the D/A that are used to latch data into the part.

### D/A pipeline delay

The AD 9767 part does not have an internal pipeline delay. However data is first strobed into the latches internal to the part, and then this value is clocked onto the D/A. This will result in a pipeline delay.

The Hardware Interface Layer first registers data from the FPGA (using an IOB to guarantee timing needs are met). The DAC pins are connected directly to the FPGA without any buffering. The Hardware Interface Layer drives the WRT and TRIG inputs of the DAC with the same clock used to register the data in the IOB. So on the second rising clock edge the data will be latched into the DAC component, and on the third rising clock edge the DAC will start top convert the digital value to an analogue level.

This means there is a pipeline of three clocks from the users Data bus in the FPGA to the analogue output.

### D/A Output Noise

The noise at the D/A outputs is mainly determined by pick up from the high speed digital signals used on the IO2V rather than the noise specified for the D/A converters or the buffer amplifier. The total noise signal, as observed on a 500MHz oscilloscope using a short 50Ohm coax cable terminated in 50Ohms, can be split into two categories, general background noise and transients. The main body of the background noise lies in 4milliVolt band, while the transient spikes may be as large as 18milliVolts.

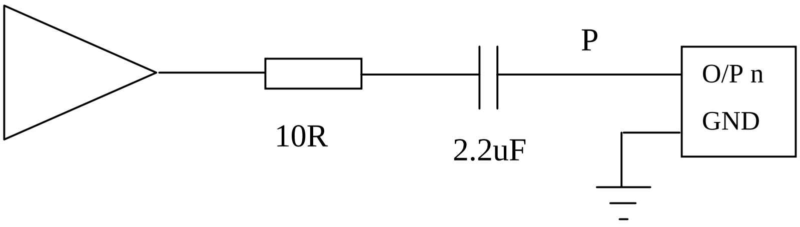
A better way of evaluating this output noise is with a spectrum analyser and the D/A output continuously updated at a 50MHz rate to a constant mid range value. Out to 150MHz this showed 50MHz and its harmonics with a maximum value of –70dB (@150MHz) re full scale output. All the remaining noise was below –75dB re full scale output.

### Analogue output options

There only build options for the analogue outputs of the HERON-IO2 is to choose AC or DC coupling. No impedance options are offered as higher impedance reduces rise times and hence output bandwidth. The standard output can drive any load impedance.

#### Standard A/C coupled

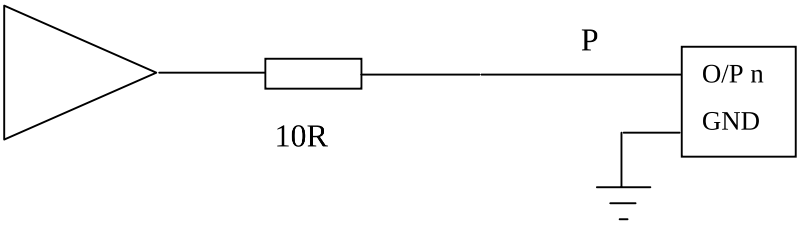
The standard output configuration is to be A/C coupled, with a 10R impedance. This results in a signal bandwidth of 1250Hz to 145MHz into 50 Ohms.



The point P is protected against overvoltage (>+5v) and undervoltage (<-5v) with respect to the HERON-IO2 ground. It is also protected against static discharge.

### D/C coupled

The D/C coupled output configuration has a 10R impedance, and a signal bandwidth of 0 to 145Mhz.



The point P is protected against overvoltage (>+5v) and undervoltage (<-5v) with respect to the HERON-IO2 ground. They are also protected against static discharge.

## Short Circuit protection

The 10R in series with the output, is sufficient to protect against a short circuited output for an indefinite time.

## ESD protection

The analogue inputs are protected against Electro Static Discharge and over voltage The devices used are Harris SP723 parts. The connections are made at the points labelled P+ and P- on the drawings in the sections above.

This protects the inputs to IEC1004-2 level 4, and provides over voltage limiting to the range 0 to +5V.

Users should ensure that cabling used in a system enables compliance with EMC directives.

### Cabling precautions

In order to achieve best signal performance, and to achieve compliance with EMC regulations for radiation and susceptibility, the cabling used in a system that uses the HERON-IO2 must be carefully designed.

The use of twisted pair cabling minimises reception and radiation of noise. This cabling has a typical characteristic impedance of around 200R. However if the HERON-IO2 has a 200R output impedance the capacitance of the cable etc causes the edges to be slowed, i.e. the output impedance along with the cable act like a low pass filter. For this reason the standard output impedance is deliberately set low, having the additional advantage of allowing the full output voltage range to be used.

The twisted pair cabling needs to be individually screened, to help further with noise performance. This screen should NOT be connected to signal ground or the 0V of the power supply, but MUST be connected to the earthed casing of your system.

In accordance with this any cabling that is provided by HUNT ENGINEERING for a HERON-IO2 will use twisted pair cables, with an individual screen for each pair. Normally the cables will be connected to a D type connector on a panel. The screens from the pairs will be connected together, and connected to the panel where the D type connector is mounted AND to a pin of the d type connector. This allows a mating cable to be used that has a screened shell, but rather than rely on the screwlocks (that can become loose and hence high resistance) to make the connection of the screen to the case, the screen can also be connected using the connector pin provided.

## Digital I/O Connector

The Digital I/O connector provides the possibility to have some digital I/Os connected directly to the FPGA, along with some serial I/O.

### I/O characteristics

The characteristics of the Digital I/O are governed by what is programmed into the FPGA. However only certain formats are possible with each voltage level on the Vcco pins of an I/O bank.

On the HERON-IO2V, some module specific LVTTL inputs are connected to banks 0 and 1 that require a 3.3V Vcco. This precludes the setting of any other value for the Vcco0 and Vcco1, so the formats supported by the Digital I/O connectors are those supported by the Virtex-II FPGA with a Vcco of 3.3V

NOTE VIRTEX II I/Os are not 5v tolerant!

## Using Digitally Controlled Impedance (DCI)

The Virtex II architecture allows the use of DCI to control the impedance of certain I/O pins. Each bank of the FPGA uses a pair of resistors connected to the VRN and VRP. FPGA that the FPGA uses to set the impedance of multiple drivers and receivers.

To use DCI the appropriate buffer must be placed in the FPGA design.

On the HERON-IO2V there are 47R 1% resistors connected to the VRN and VRP pins of bank1. This means that the I/Os I/O0 to I/O7 can use DCI without changes to the board.

## “DIGITAL I/O” Connector type

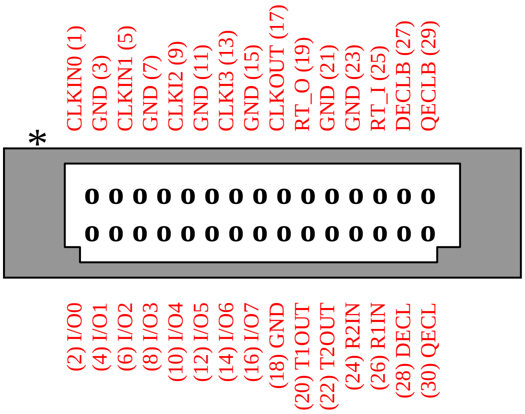
The Digital I/O connector is a surface mount 1.25mm pitch connector. It is arranged as 15 pins in each of 2 rows. It is supplied by Hirose and its part number is DF13-30DP-1.25V(50). This connector has polarisation against incorrect insertion and mechanical retention of the mating half.

The mating connector is also supplied by Hirose and has part number DF13-30DS-1.25C which requires crimp contacts part number DF13-2630SCFA. These crimps are only available from Hirose in large quantities and require special tooling. Usually if you have explained at the time of ordering how you will be using your HERON-IO2 module there will be cabling supplied that suits your needs.

If your requirements change then HUNT ENGINEERING will be able to supply assemblies or component parts to meet your needs but a charge will apply.

## “DIGITAL I/O” Connector Pin out

The connector sits against the top surface of the PCB, facing upwards away from the board.



### Digital I/Os

The digital I/Os are connected directly to I/O pins of the FPGA.

Signals CLKIN0, CLKIN1 AND CLKOUT are connected directly to the FPGA, but signals CLKI2 and CLKI3 are terminated in 220/330R and buffered prior to the FPGA.

### Serial I/Os

The serial I/Os have different uses depending on how the MAX3160 part is configured by your FPGA program.

To use these serial I/Os you must place the correct UART logic in your FPGA design – the MAX3160 is just a level converter component.

The signals T1OUT, T2OUT, R1IN and R2IN are the RS232/485 signals from the MAX3160. A and B indicate which of the two MAX3160s that the signal is associated with.

The signals RT_O and RT_I are connected to the end of 120R termination resistors. The other ends of those resistors are connected to the T2OUT and R2IN pins of the

MAX3160s. These are provided so that is termination is required for RS484 connections they can be added by correct connection of the cabling. I.e. connecting RT_O to the T1OUT_A signal provides a 120R termination for the transmit pair and connecting RT_I to the R1IN signal provides a 120R termination for the receive pair.

The signals QECL and QECLB form a differential ECL output pair which are controlled by the signal DTTL from the FPGA.

The signals DECL and DECLB form a differential ECL input pair which are decoded onto the signal QTTL on the FPGA.

## Use of the MAX3160

Of course the best way to determine how to configure the MAX3160 is to look at the data sheet provided by the manufacturer Maxim Integrated Maxim Integrated Products, found at http://www.maxim-ic.com .

There is a MAX3160 component fitted to the HERON-IO2, powered from 3.3V, with the RS232/485 signals connected to the Digital I/O Connector described above. The logic side of the part is connected directly to pins of the FPGA.

The MAX3160 has 2 digital inputs T1IN and T2IN, and 2 digital outputs R1OUT and R2OUT.

There are also three control signals also connected to the FPGA, FAST, RS485/RS232 and HDPLX.

### RS232

Set FAST = high, RS485/RS232 = low and HDPLX = low.

Now the signal T1IN is driven by the FPGA, and the RS232 version of this signal appears on the T1OUT pin of the connector.

The signal T2IN is driven by the FPGA, and the RS232 version of this signal appears on the T2OUT pin of the connector.

The signal R1OUT is driven by the MAX3160, according to the RS232 version on the R1IN pin of the connector.

The signal R2OUT is driven by the MAX3160, according to the RS232 version on the R2IN pin of the connector.

Then a UART circuit configured into the FPGA can use this as either a TX/RX + CTS/RTS RS232 channel using flow control, or alternatively 2 independent TX/RX RS232 channels without flow control.

The MAX3160 can support baud rates up to 1Mbit/second in RS232 mode. If the FAST pin is set to logic low it limits the slew rate of the signals giving better immunity to errors but then the baud rate is limited to 250Kbps.

### Full Duplex R485/RS422

Set FAST = high, RS485/RS232 = high and HDPLX = low.

Now the signal T1IN is driven by the FPGA, and the inverted RS422/RS485 version of this signal appears on the T1OUT pin of the connector, the non-inverted version of this signal appears on the T2OUT pin of the connector.

The signal T2IN is driven by the FPGA, and is used as a driver enable signals that is driven high to enable the outputs T1OUT and T2OUT.

The signal R1OUT is not used.

The signal R2OUT is driven by the MAX3160, according to the signal formed by the RS485/RS422 pair on the pins R1IN (inverting) and R2IN (non-inverting) pins of the connector.

Then a UART circuit configured into the FPGA can use this as a TX/RX signal pair that are driven differentially according to RS422/RS485.

The MAX3160 can support baud rates up to 10Mbit/second in R485 mode. If the FAST pin is set to logic low it limits the slew rate of the signals giving better immunity to errors but then the baud rate is limited to 250Kbps.

### Half Duplex R485/RS422

Set FAST = high, RS485/RS232 = high and HDPLX = high.

Now the signal T1IN is driven by the FPGA, and the inverted RS422/RS485 version of this signal appears on the T1OUT pin of the connector, the non-inverted version of this signal appears on the T2OUT pin of the connector.

The signal T2IN is driven by the FPGA, and is used as a driver enable signals that is driven high to enable the outputs T1OUT and T2OUT.

The signal R1OUT is not used.

The signal R2OUT is driven by the MAX3160, according to the signal formed by the RS485/RS422 pair on the pins T1OUT (inverting) and T2OUT (non-inverting) pins of the connector.

Then a UART circuit configured into the FPGA can use this as a TX/RX signal pair that are driven differentially according to RS422/RS485 on a single pair formed by T1OUT and T2OUT. The direction of Transmit/Receive is now controlled using the Driver enable connected to T2IN. This type of RS485 connection is often used for multi-drop applications where all devices, except one, default to input and only "respond" when polled by the "master".

The MAX3160 can support baud rates up to 10Mbit/second in R485 mode. If the FAST pin is set to logic low it limits the slew rate of the signals giving better immunity to errors but then the baud rate is limited to 250Kbps.

## ESD protection

All of the Digital I/Os are protected against Electro Static Discharge and over voltage The devices used are Harris SP723 parts.

This protects the inputs to IEC1004-2 level 4, and provides over voltage limiting to the range 0 to +5V.

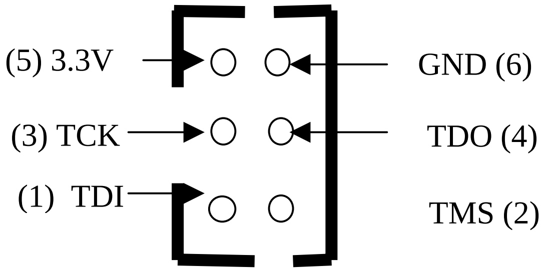
For the Serial I/O the MAX3160 provides protection against wiring faults etc, and the use of slew rate limiting is provided to minimise radiated noise. Users should ensure that cabling used in a system enables compliance with EMC directives.

## The JTAG programmable Configuration PROM

The module has been built with a JTAG programmable FLASH based PROM for the user FPGA, so that your FPGA design can be programmed into this PROM. Then the user FPGA can be configured with your FPGA design on power up, and re-configuration can be forced using the send bitstream command over HSB with a zero length bitstream.

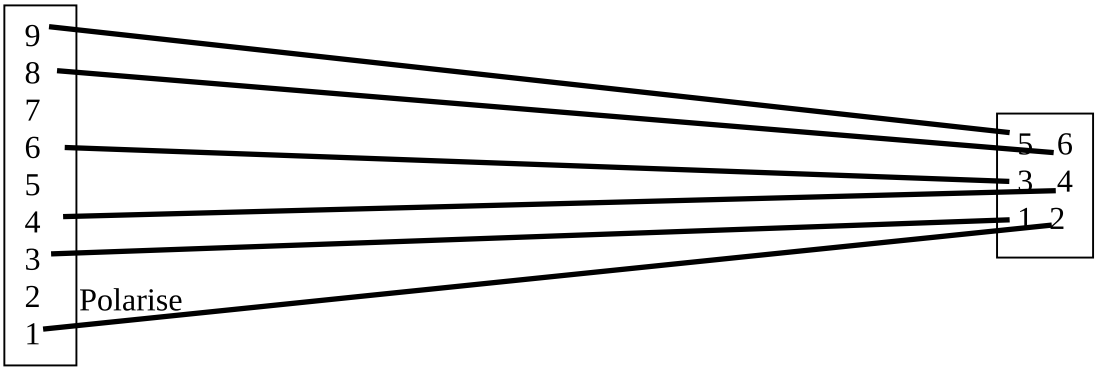
The module will have a JTAG connector fitted that is a Hirose 2mm 3x2 connector of part number DF11-6DP-DSA(01). The mating half that is required for cabling is also a Hirose part, the housing is DF11-6DS-2C and crimp part number DF11-2428CSA.

The pinout is:-



Top view of connector.

The cable supplied with modules that have the PROM option fitted, is as follows:-



Which can be fitted to a Xilinx JTAG cable part number HW-JTAG-PC and used to connect to the PROM on the module.

The JTAG chain from the JTAG connector is linked to the PROM, and also to the Virtex II.. With the JTAG chain linked to the Virtex II this makes it possible to download the FPGA design directly via JTAG, it also allows software packages such as Chipscope to be used to help debug the design in the FPGA.

## User FPGA Boot from PROM Jumper

If the option of booting the user FPGA from ROM is fitted, the jumper can be used to select if the FPGA boots from the ROM ,or via HSB or directly via JTAG.

For normal operation (HSB) the jumper should NOT be fitted.

## Uncommitted Module Interconnects

There are some "Uncommitted Interconnect" signals defined by the HERON specification, which are simply connected to all modules.

These are intended to connect control signals between modules, for example a processor module can (via software) drive one of these signals with one of is timer outputs. Then if an I/O module can accept its clock input from one of these signals, it is possible to implement a system with a programmable clock. There will be other uses for these signals that are module design dependent.

The HERON-IO2 connects these signals to FPGA I/O pins allowing the user configuration to use these if required.

## General Purpose LEDs

There are some general purpose LEDS on the HERON-IO2, which are driven by buffers that are external to the FPGA.

The LEDs labelled 0 to 4 are driven by the FPGA pins LED0 to LED4 respectively. The LED will illuminate when the FPGA drives the pin low.

## Other HERON module signals

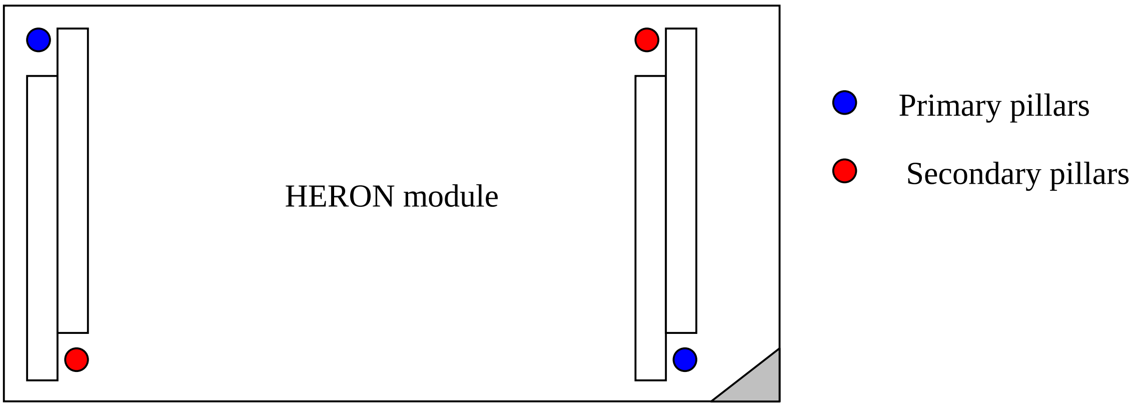
There are many signals that are connected between the FPGA on the HERON-IO2 and the HERON module connectors. Most of these signals will only be used by advanced users of the HERON-IO2. The FPGA pinning of these signals is shown in the appendix of this manual, and their uses in a system is described in the HERON module specification found on the HUNT ENGINEERING CD and Web Site.

The FPGA sets any I/O pins of the device that are not listed in the design to have a 50-150K pull down. Most of the HERON module signals are pulled to their inactive state by 10K resistors so this 50K will have no effect. However the UDPRES signal does not, and setting this signal low will cause your whole board to be reset. Thus it is important that the UDPRES pin is driven high by the FPGA if it is not being used.

It is also advised to do the same with the LED pins to prevent them becoming illuminated erroneously.

Fitting HERON modules to your carrier is very simple. Ensure that the module carrier does NOT have power applied when fitting modules, and normal anti-static precautions should be followed at all times.

Each HERON slot has four positions for fixing pillars

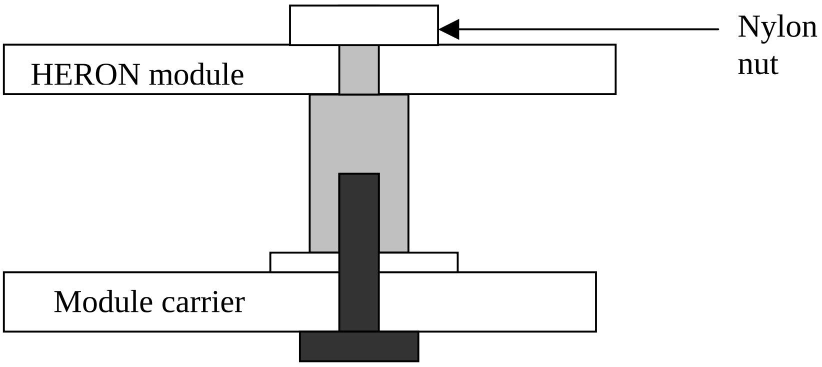


The Carrier card will probably only have spacing pillars fitted to the primary location for each HERON slot. The pillars for the secondary locations will be supplied as an accessory. The reason for this is that the legacy GDIO modules cannot be fitted if the secondary pillars are in place.

The HERON modules are asymmetric about their connectors, so if a module is fitted entirely the wrong way round, the module does not line up with the markings on the carrier card. In particular, notice the triangles on the silk screen of the HERON modules and the HERON slots of the carrier card. These should be overlaid when the module is fitted.

The HERON connectors are polarised, preventing incorrect insertion. So if more than a gentle force is needed to push the module home, check to make sure that it is correctly aligned. Take care not to apply excessive pressure to the centre of the module as this could stress the module's PCB unnecessarily.

Normally the primary fixings will be enough to retain the modules, simply fit the nylon bolts supplied in the accessory kit to the top thread of each mounting pillar.



If the environment demands, the secondary fixing pillars can be fitted to modules that allow their use.

# Achievable System Throughput

In a HERON system there are many factors that can affect the achievable system throughput. It must be remembered at all times that the part of the system that has the lowest limit on bandwidth will govern the throughput of the system.

The HERON-IO2 can access the HERON carriers FIFOs in 32 bits mode. It can (with the right contents) transfer one 32bit word in and another out in the same clock cycle.

For example running at a FIFO clock speed of 50Mhz, the HERON-IO2 can transfer 200Mbytes/sec in at the same time as transferring 200Mbytes/sec out.

The use of faster clock speeds for the FIFOs will of course result in higher data rates.

# Troubleshooting

The following sections attempt to cover all likely problems. Please check through this section before contacting technical support.

## Hardware

If the Hardware has been installed according to the Instructions there is very little that can be wrong.

- Has the "DONE" LED gone out – if not then the FPGA is not configured
- Perhaps you do not have a FIFO clock being driven from your FPGA Design
- Not driving the UDPRES signal high in your FPGA Design will result in unpredictable behaviour.

## Software

As long as the software has been installed using the installation program supplied on the HUNT ENGINEERING CD, there should be little problem with the software installation.

If you have problems then return to one of the example programs supplied with the system.

HUNT ENGINEERING have performed testing on its products to ensure that it is possible to comply with the European CE marking directives. The HERON-IO2 cannot be CE marked as it is a component in a system, but as long as the following recommendations are followed, a system containing the HERON-IO2 could be CE marked.

The immense flexibility of the HUNT ENGINEERING product range means that individual systems should be marked in accordance with the directives after assembly.

1.The host computer or housing in which the HERON-IO2 is installed is properly assembled with EMC and LVD in mind and ideally should itself carry the CE mark.

2. Any cabling between boards or peripherals is either entirely inside the case of the host computer, or has been assembled and tested in accordance with the directives.

The HERON-IO2 digital I/Os ARE protected against Static discharge, so if the cabling does exit the case, there is suitable protection already fitted.

HUNT ENGINEERING are able to perform system integration in accordance with these directives if you are unsure of how to achieve compliance yourself.

# Technical Support

Technical support for HUNT ENGINEERING products should first be obtained from the comprehensive Support section http://www.hunteng.co.uk/support/index.htm on the HUNT ENGINEERING web site. This includes FAQs, latest product, software and documentation updates etc. Or contact your local supplier - if you are unsure of details please refer to http:/www.hunteng.co.uk for the list of current re-sellers.

HUNT ENGINEERING technical support can be contacted by emailing support@hunteng.co.uk, calling the direct support telephone number +44 (0)1278 760775, or by calling the general number +44 (0)1278 760188 and choosing the technical support option.

# Appendix 1 – HERON Serial Bus Commands

## Module address

The HERON-IO2 is configured to respond to Heron Serial Bus (HSB) commands addressed to it using the combination of the Board number and slot number that the module is fitted to. In this way multiple HERON-FPGA modules can be uniquely addressed in the same system. The HSB address is a 7 bit address that is formed by the bottom three bits of the slot number ( slots 1 to 4 are valid – 001,010,011, 100) with the 4 bits from the board number switch forming the top 4 bits of the seven.

e.g. on board number 1 slot 2 the address would be (board number<<3) || slot[2.0] which is 0x06.

The HERON-IO2 can respond to three different types of serial bus commands:-

## Module Enquiry

The HERON-IO2 can receive a message requesting its module type:-

Master to FPGA module

module type query (01)-->address of requestor

It will then send a reply as follows :-

FPGA module to "original master"

module query response(02)-->module address (from)-->module type (04)

-->family number(02)-->option(2)-->String byte 0 -->String byte1….String byte26

The string is the string that Xilinx put into their bitstream files. It is always 27 bytes long, but can actually be null terminated before that. e.g. a Spartan II -5 would return 2s200fg456-5.

## FPGA Configuration

The Configuration transaction will be:-

Master to FPGA module

Configure (03)-->address of requestor--> first config byte-->

2nd config byte......last config byte

After which the FPGA module will reply:-

FPGA module to original master

configuration success (05)/configuration fail(06)-->module address

## User I/O

Any further use of the HSB will be defined by the bitstream supplied to the module.

The actual use of these messages cannot be defined here, but the format of them must be:-

Master to FPGA module

user write (08) -->address of requestor -->register address byte -->value byte

-->optional value byte-->optional value byte......

In this way single or multiple bytes can be written, starting from the address given.

Nothing is returned from a write request.

This will result in the 8 bit address being written into the application FPGA using an address strobe, then one or more data bytes being written to the application FPGA using a data strobe, and qualified by a write signal. It is therefore the responsibility of the application FPGA to support auto incrementing addresses if required by its function.

For a read request

Master to FPGA module

user read (09) -->address of requestor -->register address byte -->length byte

In this way single or multiple bytes can be requested, starting from the address given.

The reply will be

FPGA module to original master

user read response(10)-->module address-->data byte-->optional data byte

This will result in the 8 bit address being written into the application FPGA using an address strobe, then one or more data bytes being read from the application FPGA using a data strobe, and qualified by the absence of write signal.

# Appendix 2 – FPGA Pinout for development tools

The following is the pin out of the FPGA, so that the signals can be connected in the Xilinx development tools.

Data Out FIFO:

| Signal name | FPGA pin | Description |
|---|---|---|
| DO0 | U19 | HERON FIFO Data bit output |
| DO1 | T18 | HERON FIFO Data bit output |
| DO2 | W22 | HERON FIFO Data bit output |
| DO3 | U21 | HERON FIFO Data bit output |
| DO4 | T20 | HERON FIFO Data bit output |
| DO5 | T21 | HERON FIFO Data bit output |
| DO6 | R18 | HERON FIFO Data bit output |
| DO7 | U22 | HERON FIFO Data bit output |
| DO8 | R19 | HERON FIFO Data bit output |
| DO9 | P18 | HERON FIFO Data bit output |
| DO10 | R22 | HERON FIFO Data bit output |
| DO11 | P21 | HERON FIFO Data bit output |
| DO12 | P22 | HERON FIFO Data bit output |
| DO13 | N18 | HERON FIFO Data bit output |
| DO14 | N21 | HERON FIFO Data bit output |
| DO15 | M17 | HERON FIFO Data bit output |
| DO16 | M19 | HERON FIFO Data bit output |
| DO17 | M18 | HERON FIFO Data bit output |
| DO18 | L20 | HERON FIFO Data bit output |
| DO19 | L17 | HERON FIFO Data bit output |
| DO20 | L21 | HERON FIFO Data bit output |
| DO21 | L22 | HERON FIFO Data bit output |
| DO22 | K21 | HERON FIFO Data bit output |
| DO23 | K22 | HERON FIFO Data bit output |
| DO24 | J21 | HERON FIFO Data bit output |
| DO25 | J18 | HERON FIFO Data bit output |
| DO26 | J22 | HERON FIFO Data bit output |
| DO27 | H19 | HERON FIFO Data bit output |
| DO28 | H18 | HERON FIFO Data bit output |
| DO29 | G21 | HERON FIFO Data bit output |
| DO30 | G18 | HERON FIFO Data bit output |
| DO31 | G20 | HERON FIFO Data bit output |
| | | |
| F0CLK | D2 | O/P FIFO Clock output to input of buffer. Use to drive correct frequency. |
| DOCLK/GCLKx | W12 | O/P FIFO Clock output of buffer – use with DLL for internal logic |
| | | |
| DOF0CONT0 | Y14 | O/P FIFO #0 Write enable (active high) output |
| DOF0CONT1 | AB17 | O/P FIFO #0 Full Flag (active low) input |
| DOF0CONT2 | AB16 | O/P FIFO #0 Almost full flag (active low) input |
| | | |
| DOF1CONT0 | W15 | O/P FIFO #1 Write enable (active high) output |
| DOF1CONT1 | V15 | O/P FIFO #1 Full Flag (active low) input |
| DOF1CONT2 | AB18 | O/P FIFO #1 Almost full flag (active low) input |
| | | |
| DOF2CONT0 | AB15 | O/P FIFO #2 Write enable (active high) output |

| Signal name | FPGA pin | Description |
|---|---|---|
| DOF2CONT1 | AA13 | O/P FIFO #2 Full Flag (active low) input |
| DOF2CONT2 | Y17 | O/P FIFO #2 Almost full flag (active low) input |
| | | |
| DOF3CONT0 | W17 | O/P FIFO #3 Write enable (active high) output |
| DOF3CONT1 | Y16 | O/P FIFO #3 Full Flag (active low) input |
| DOF3CONT2 | V16 | O/P FIFO #3 Almost full flag (active low) input |
| | | |
| DOF4CONT0 | AA15 | O/P FIFO #4 Write enable (active high) output |
| DOF4CONT1 | AA17 | O/P FIFO #4 Full Flag (active low) input |
| DOF4CONT2 | W21 | O/P FIFO #4 Almost full flag (active low) input |
| | | |
| DOF5CONT0 | Y15 | O/P FIFO #5 Write enable (active high) output |
| DOF5CONT1 | W16 | O/P FIFO #5 Full Flag (active low) input |
| DOF5CONT2 | U20 | O/P FIFO #5 Almost full flag (active low) input |

These FIFO signals should be used via the library symbol supplied by HUNT ENGINEERING and are only mentioned here for completeness.

Data In FIFO:

| Signal name | FPGA pin | Description |
|---|---|---|
| DI0 | V8 | HERON FIFO Data bit input |
| DI1 | AA5 | HERON FIFO Data bit input |
| DI2 | AB5 | HERON FIFO Data bit input |
| DI3 | Y7 | HERON FIFO Data bit input |
| DI4 | W6 | HERON FIFO Data bit input |
| DI5 | Y6 | HERON FIFO Data bit input |
| DI6 | V7 | HERON FIFO Data bit input |
| DI7 | Y2 | HERON FIFO Data bit input |
| DI8 | W2 | HERON FIFO Data bit input |
| DI9 | U4 | HERON FIFO Data bit input |
| DI10 | Y1 | HERON FIFO Data bit input |
| DI11 | V2 | HERON FIFO Data bit input |
| DI12 | W1 | HERON FIFO Data bit input |
| DI13 | T4 | HERON FIFO Data bit input |
| DI14 | T3 | HERON FIFO Data bit input |
| DI15 | U2 | HERON FIFO Data bit input |
| DI16 | V1 | HERON FIFO Data bit input |
| DI17 | U1 | HERON FIFO Data bit input |
| DI18 | T2 | HERON FIFO Data bit input |
| DI19 | R4 | HERON FIFO Data bit input |
| DI20 | T1 | HERON FIFO Data bit input |
| DI21 | P3 | HERON FIFO Data bit input |
| DI22 | P5 | HERON FIFO Data bit input |
| DI23 | P4 | HERON FIFO Data bit input |
| DI24 | N5 | HERON FIFO Data bit input |
| DI25 | N4 | HERON FIFO Data bit input |
| DI26 | N3 | HERON FIFO Data bit input |
| DI27 | N2 | HERON FIFO Data bit input |
| DI28 | M5 | HERON FIFO Data bit input |
| DI29 | N1 | HERON FIFO Data bit input |
| DI30 | M6 | HERON FIFO Data bit input |
| DI31 | M1 | HERON FIFO Data bit input |
| | | |
| F1CLK | E1 | I/P FIFO Clock output to input of buffer. Use to drive correct frequency. |
| DICLK/GCLKx | Y11 | I/P FIFO Clock output of buffer – use with DLL for internal logic |
| | | |

| | | |
|---|---|---|
| DIF0CONT0 | AB6 | I/P FIFO #0 Read enable (active high) output |
| DIF0CONT1 | AB13 | I/P FIFO #0 output enable (active low) output |
| DIF0CONT2 | V9 | I/P FIFO #0 Empty Flag (active low) input |
| DIF0CONT3 | V11 | I/P FIFO #0 Almost Empty flag (active low) input |
| | | |
| DIF1CONT0 | AA7 | I/P FIFO #1 Read enable (active high) output |
| DIF1CONT1 | AA18 | I/P FIFO #1 output enable (active low) output |
| DIF1CONT2 | AB9 | I/P FIFO #1 Empty Flag (active low) input |
| DIF1CONT3 | W11 | I/P FIFO #1 Almost Empty flag (active low) input |
| | | |
| DIF2CONT0 | W7 | I/P FIFO #2 Read enable (active high) output |
| DIF2CONT1 | Y13 | I/P FIFO #2 output enable (active low) output |
| DIF2CONT2 | V10 | I/P FIFO #2 Empty Flag (active low) input |
| DIF2CONT3 | U11 | I/P FIFO #2 Almost Empty flag (active low) input |
| | | |
| DIF3CONT0 | W8 | I/P FIFO #3 Read enable (active high) output |
| DIF3CONT1 | V13 | I/P FIFO #3 output enable (active low) output |
| DIF3CONT2 | AA9 | I/P FIFO #3 Empty Flag (active low) input |
| DIF3CONT3 | Y12 | I/P FIFO #3 Almost Empty flag (active low) input |
| | | |
| DIF4CONT0 | Y8 | I/P FIFO #4 Read enable (active high) output |
| DIF4CONT1 | W14 | I/P FIFO #4 output enable (active low) output |
| DIF4CONT2 | W10 | I/P FIFO #4 Empty Flag (active low) input |
| DIF4CONT3 | AA12 | I/P FIFO #4 Almost Empty flag (active low) input |
| | | |
| DIF5CONT0 | AA8 | I/P FIFO #5 Read enable (active high) output |
| DIF5CONT1 | V14 | I/P FIFO #5 output enable (active low) output |
| DIF5CONT2 | Y10 | I/P FIFO #5 Empty Flag (active low) input |
| DIF5CONT3 | W13 | I/P FIFO #5 Almost Empty flag (active low) input |

These FIFO signals should be used via the library symbol supplied by HUNT ENGINEERING and are only mentioned here for completeness.

Analog I/P (A/Ds)

| Signal name | FPGA pin | Description |
|---|---|---|
| AD_A0 | D7 | Data input from A/D A |
| AD_A1 | C7 | Data input from A/D A |
| AD_A2 | D8 | Data input from A/D A |
| AD_A3 | C8 | Data input from A/D A |
| AD_A4 | B7 | Data input from A/D A |
| AD_A5 | A7 | Data input from A/D A |
| AD_A6 | D9 | Data input from A/D A |
| AD_A7 | C9 | Data input from A/D A |
| AD_A8 | B8 | Data input from A/D A |
| AD_A9 | A8 | Data input from A/D A |
| AD_A10 | B9 | Data input from A/D A |
| AD_A11 | A9 | Data input from A/D A |
| OTRA | B10 | A/D A Over range pin (high means input signal is too large) |
| | | |
| AD_B0 | C12 | Data input from A/D B |
| AD_B1 | B12 | Data input from A/D B |
| AD_B2 | E12 | Data input from A/D B |
| AD_B3 | D12 | Data input from A/D B |
| AD_B4 | C13 | Data input from A/D B |
| AD_B5 | D13 | Data input from A/D B |
| AD_B6 | C14 | Data input from A/D B |
| AD_B7 | D14 | Data input from A/D B |
| AD_B8 | A15 | Data input from A/D B |
| AD_B9 | B15 | Data input from A/D B |
| AD_B10 | E13 | Data input from A/D B |

| | | |
|---|---|---|
| AD_B11 | E14 | Data input from A/D B |
| OTRB | C15 | A/D B Over range pin (high means input signal is too large) |
| | | |

| Signal name | FPGA pin | Description |
|---|---|---|
| PCLK+ | A11 | B0 I/O96P GCLK4S : LVPECL+ I/P FROM ‘AC CLK’ |
| PCLK- | B11 | B0 I/O96N GCLK5P : LVPECL- I/P FROM ‘AC CLK’ |
| ENC_A+ | B16 | B1 I/O22P : A/D ‘A’ LVPECL+ SAMPLE CLOCK |
| ENC_A- | A16 | B1 I/O22N : A/D ‘A’ LVPECL- SAMPLE CLOCK |
| ENC_A+ | D17 | B1 I/O4P : A/D ‘A’ LVPECL+ SAMPLE CLOCK |
| ENC_A- | C17 | B1 I/O4N : A/D ‘A’ LVPECL- SAMPLE CLOCK |
| ENCODE_B+ | D22 | B2 I/O3P : A/D ‘B’ LVPECL+ SAMPLE CLOCK |
| ENCODE_B- | D21 | B2 I/O3N : A/D ‘B’ LVPECL- SAMPLE CLOCK |
| ENCODE_B+ | F22 | B2 I/O21P : A/D ‘B’ LVPECL+ SAMPLE CLOCK |
| ENCODE_B- | F21 | B2 I/O21N : A/D ‘B’ LVPECL- SAMPLE CLOCK |

Analog O/P (D/As)

| Signal name | FPGA pin | Description |
|---|---|---|
| DA_A0 | M3 | Data output to D/A A |
| DA_A1 | M4 | Data output to D/A A |
| DA_A2 | P1 | Data output to D/A A |
| DA_A3 | P2 | Data output to D/A A |
| DA_A4 | R1 | Data output to D/A A |
| DA_A5 | R2 | Data output to D/A A |
| DA_A6 | R5 | Data output to D/A A |
| DA_A7 | T5 | Data output to D/A A |
| DA_A8 | V3 | Data output to D/A A |
| DA_A9 | V4 | Data output to D/A A |
| DA_A10 | Y9 | Data output to D/A A |
| DA_A11 | W9 | Data output to D/A A |
| DA_A12 | AB10 | Data output to D/A A |
| DA_A13 | AA10 | Data output to D/A A |
| TRIG0 | E4 | Clock for D/A A |
| WRT0 | C6 | Data Strobe for D/A A |
| DA_B0 | B5 | Data output to D/A B |
| DA_B1 | A5 | Data output to D/A B |
| DA_B2 | D10 | Data output to D/A B |
| DA_B3 | C10 | Data output to D/A B |
| DA_B4 | E11 | Data output to D/A B |
| DA_B5 | F11 | Data output to D/A B |
| DA_B6 | E7 | Data output to D/A B |
| DA_B7 | F5 | Data output to D/A B |
| DA_B8 | G5 | Data output to D/A B |
| DA_B9 | G4 | Data output to D/A B |
| DA_B10 | G3 | Data output to D/A B |
| DA_B11 | H2 | Data output to D/A B |
| DA_B12 | H1 | Data output to D/A B |
| DA_B13 | J4 | Data output to D/A B |
| TRIG1 | C1 | Clock for D/A B |
| WRT1 | E8 | Data Strobe for D/A B |

Digital IO on Connectors

| Signal name | FPGA pin | Description |
|---|---|---|
| IO0 (L54N_1) | A14 | General purpose I/O with selectable I/O format |

| | | |
|---|---|---|
| IO1 (L54P_1) | B14 | General purpose I/O with selectable I/O format |
| IO2 (L05N_1) | A17 | General purpose I/O with selectable I/O format |
| IO3 (L05P_1) | B17 | General purpose I/O with selectable I/O format |
| IO4 (L93N_1) | A13 | General purpose I/O with selectable I/O format |
| IO5 (L93P_1) | B13 | General purpose I/O with selectable I/O format |
| IO6 (L21N_1) | C16 | General purpose I/O with selectable I/O format |
| IO7 (L21P_1) | D16 | General purpose I/O with selectable I/O format |

Clocks

| Signal name | FPGA pin | Description |
|---|---|---|
| OSC0 | C5 | User Oscillator input from socketed Xtal Osc |
| OSC1 | B6 | User Oscillator input from socketed Xtal Osc |
| OSC2 | E9 | User Oscillator input from surface mount Xtal Osc |
| OSC3 | E10 | User Oscillator input from surface mount Xtal Osc |
| | | |
| CLKIN0 | F4 | User Clock input from connector (unbuffered) |
| CLKIN1 | F3 | User Clock input from connector (unbuffered) |
| CLKI2 | B4 | User Clock input from connector (buffered) |
| CLKI3 | A4 | User Clock input from connector (buffered) |
| | | |
| CLKOUT | E3 | User Clock output to connector (unbuffered) |
| | | |
| | | Repeated from above |
| F0CLK | D2 | O/P FIFO Clock output to input of buffer. Use to drive correct frequency. |
| DOCLK/GCLKx | W12 | O/P FIFO Clock output of buffer – use with DLL for internal logic |
| F1CLK | E1 | I/P FIFO Clock output to input of buffer. Use to drive correct frequency. |
| DICLK/GCLKx | Y11 | I/P FIFO Clock output of buffer – use with DLL for internal logic |

LEDs

| Signal name | FPGA pin | Description |
|---|---|---|
| LED0 | H3 | General Purpose LED |
| LED1 | J1 | General Purpose LED |
| LED2 | L5 | General Purpose LED |
| LED3 | J5 | General Purpose LED |
| LED4 | J2 | General Purpose LED |

Control connections to HERON connectors

| Signal name | FPGA pin | Description |
|---|---|---|
| ADDR/FLAGSEL | F12 | Selector driven by Carrier board to determine the function of the almost flags |
| BOOTEN | G1 | This module can drive this low if it wishes the carrier to operate regardless of the Config signal |
| UDPRES | H4 | This module can drive this to reset the carrier YOU MUST DRIVE THIS SIGNAL HIGH IF NOT USING IT! |
| RESET | E15 | Reset input from Carrier card |
| CONFIG | E16 | Open collector signal |

Carrier and Module ID

| Signal name | FPGA | Description |
|---|---|---|

|  | pin |  |
|---|---|---|
| CID0 | K5 | Carrier ID driven by the carrier board |
| CID1 | K1 | Carrier ID driven by the carrier board |
| CID2 | K3 | Carrier ID driven by the carrier board |
| CID3 | K4 | Carrier ID driven by the carrier board |
|  |  |  |
| MID0 | K2 | Module ID driven by the carrier board |
| MID1 | L6 | Module ID driven by the carrier board |
| MID2 | L4 | Module ID driven by the carrier board |
| MID3 | L3 | Module ID driven by the carrier board |

Uncommitted Module Interconnects

| Signal name | FPGA pin | Description |
|---|---|---|
| UMI0 | E2 | Uncommitted Module interconnect |
| UMI1 | H5 | Uncommitted Module interconnect |
| UMI2 | F2 | Uncommitted Module interconnect |
| UMI3 | F1 | Uncommitted Module interconnect |

Serial options

| Signal name | FPGA pin | Description |
|---|---|---|
| T1IN_A | E20 | Serial Data Bit output from FPGA (connected to MAX3160 ) |
| T2IN_A | C22 | Serial Data Bit output from FPGA (connected to MAX3160 ) |
| | | |
| R1OUT_A | A19 | Serial Data Bit input to FPGA (connected to MAX3160 ) |
| R2OUT_A | C18 | Serial Data Bit input to FPGA (connected to MAX3160 ) |
| | | |
| RS485/232_A | E21 | Control input to MAX3160 |
| HDPLX_A | F18 | Control input to MAX3160 |
| FAST_A | F20 | Control input to MAX3160 |
| | | |
| DTTL | E22 | Output from FPGA that drives Diff ECL driver |
| QTTL | F19 | Input to FPGA from Diff ECL receiver |

User interface between HSB device and FPGA (N.B. Some signals also used during configuration of FPGA.

| Signal name | VII pin | SII pin | Description |
|---|---|---|---|
| Data0 | V18 | D20 | Data Bit for read/write via HSB. |
| Data1 | V17 | H22 | Data Bit for read/write via HSB. |
| Data2 | W18 | H20 | Data Bit for read/write via HSB. |
| Data3 | Y18 | K20 | Data Bit for read/write via HSB. |
| Data4 | Y5 | N22 | Data Bit for read/write via HSB. |
| Data5 | W5 | R21 | Data Bit for read/write via HSB. |
| Data6 | AB4 | T22 | Data Bit for read/write via HSB. |
| Data7 | AA4 | Y21 | Data Bit for read/write via HSB. |
| | | | |
| Address Strobe | AA19 | V19 | Rising edge strobes the 8 bit address from the Data pins |
| Data Strobe | AA3 | C19 | Low to show an access in progress. |
| R/notW | Y4 | A20 | State of this pin when Data Strobe is active defines whether the operation is read(High) or write (low) |
| Ready | AB19 | C21 | The FPGA asserts this signal low when either :-<br>a) during a Write to the FPGA the data has been latched<br>b) during a read from the FPGA the data has been driven |

These signals should be used via the library symbol HE_USER supplied by HUNT ENGINEERING and are only mentioned here for completeness.