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
www.hunteng.co.uk
www.hunt-dsp.com

Camera Link Area-scan Camera Interface using the FPGA12

v1.0 R.Williams 14-02-06

The HERON-FPGA and HERON-IO families are ranges of HERON modules with FPGAs, often combined with some interface capability. The HERON-FPGA family in particular provides an FPGA along with a large number of signals routed to general-purpose connectors. These modules are suitable for connecting to digital cameras, where the control of the camera and image capture can be performed by the FPGA fitted to the module.

For the development of the FPGA function, HUNT ENGINEERING provides a Hardware Interface Layer written in VHDL. This layer allows developers to focus on the application-specific parts of the system. All of the module hardware should be accessed using parts from the library.

Through the Camera Link Area-scan Camera example HUNT ENGINEERING provides a structured starting point for the development of an area-scan camera interface. The example includes several components suitable for processing a stream of camera data. It is intended to be a generic tutorial, so does not address any camera specific issues.

History

Rev 1.0	Adapted from the example for HERON-FPGA9.

Concepts

FPGA devices are programmable hardware, which can be configured to meet the needs of your system. As with a microprocessor, the function of the FPGA is controlled entirely by the program – and by the peripherals the FPGA is connected to.

A HERON module has access to FIFOs for communicating with other modules in the system – there may be up to 6 input FIFOs and 6 output FIFOs. It may have additional interfaces, such as ADCs, or level-shifting buffers, allowing it to interface to the real world.

The HERON-FPGA family use Xilinx re-configurable logic devices. These can be programmed at low level, but can also be programmed using high level blocks, such as FIR filters, FFT transforms and so on. This process is covered in a separate paper.

The Camera Link Area-scan Example

Every HERON-FPGA module is provided with a Hardware Interface Layer that makes it simple for a user's FPGA design to access the hardware. Some of the HERON-FPGA modules also have a Camera Link area-scan camera interface example entitled "CamLink_Cam". For the rest of this document we will discuss the Camera Link area-scan camera example provided for the Virtex-4 FX version of the HERON-FPGA12.

Camera Link is a communication interface that was specifically developed by camera and frame-grabber manufacturers for use with vision applications. Camera Link works in one of three different configurations, Base, Medium and Full. The Camera Link example has been developed specifically for the Base Configuration only. Using this configuration however, it is possible to support cameras from 8-bit pixel resolutions up to 24-bit RGB.

Please note, in order to correctly interface to Camera Link, the FPGA must be able to connect to LVDS signals. The Virtex-4 FX FPGA IO standards include LVDS support, while the Spartan-II FPGA IO standards do not. As a result Spartan-II versions of HERON-FPGA modules cannot be used with the Camera Link example. You can refer to the later sections on 'Signal Voltage Levels' for more information on the requirements of connecting to LVDS.

The "CamLink_Cam" example is supplied as a project for ISE. It contains the top-level design, and user constraints that define the pinning of the FPGA and timing of the clocks.

The code for the camera interface is implemented in the User Application source file, user_ap1.vhd, and the files below that. The part of the design tree that includes the files hsb1.vhd and below contains a set of registers that are programmed over the HSB message interface in order to control camera interface operation. The HSB interface also provides a mechanism for transmitting and receiving RS-232 messages between the Camera Link camera and the FPGA camera interface. The part of the design tree that includes the file clink.vhd and below contains the Camera Link interface. The file camera.vhd contains generic camera processing functions. All of these files can be modified to make your own camera interface.

Although the FPGA can be used to capture and process image data without the need for a DSP, the standard IP outputs full rate image data. This cannot be sent to a Host PC without providing a buffer that takes care of the "burst" nature of PCI transactions. So for this example we actually capture images into the memory of a DSP module, then use the DSP to format that into the Windows '`.bmp`' format and write it to the hard drive of the host PC using Server/Loader functions.

So in addition to the FPGA example, there is a software example that can be run on a DSP. This program is provided as a set of two C source files and three header files. The program performs four tasks. The first task is to set up the camera interface over the HSB interface. The second task is to send and receive a series of RS-232 messages between the FPGA and camera. Thirdly an image is captured by the FPGA module, transmitted through a HERON-FIFO and received on the DSP. Finally a series of bitmap creation functions are called to create a Bitmap file (image.bmp) from the captured image.

The created file can then be viewed by host system imaging software.

This tutorial assumes that the version of ISE design tool you are using is the same as the version of the camera-link example. There are application notes on the HUNT ENGINEERING CD that describe how to re-use CD example projects with different versions of ISE if this is necessary. In addition, if you are using a different synthesis environment to ISE there is an application note 'Using VHDL tools other than ISE' that you can refer to.

What the Bit-stream Does

The Camera Link Area-scan Camera example provided on the HUNT ENGINEERING CD includes bit-streams that can be loaded directly onto the Virtex-4 FX HERON-FPGA12 module. These bit-streams implement a generic area-scan camera interface, with automatic frame-size and region of interest detection and programmable region of interest for capture, and frame capture control.

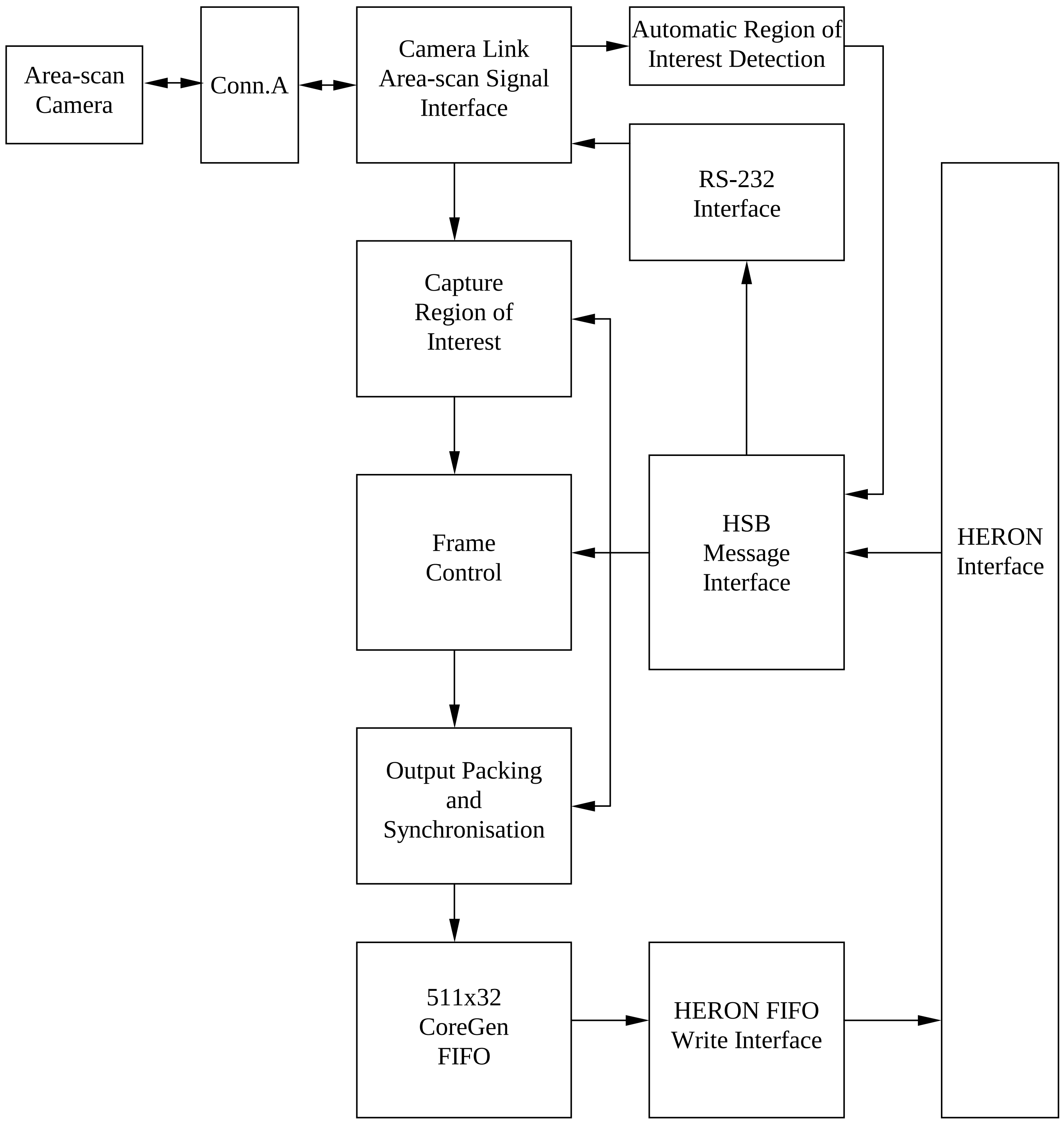
The bit-streams are provided for connecting to a camera that operates using Camera Link. There are two versions of the bit-stream. One bitstream for cameras operating at 24MHz and below and another bitstream for cameras working at 25MHz or above.

The VHDL provided in the 'CamLink_Cam' example includes source code for two different frequency Camera Link de-serializers. Each de-serializer design works with a different range of camera pixel clock frequency. The appropriate frequency range de-serializer must be used based on the pixel clock frequency of the camera you are using.

For the cameras with pixel clocks of 24MHz or below, the bitstream ending '_lf' must be used. This bitstream has been built with the Low Frequency Camera Link de-serializer. For the cameras with pixel clock of 25MHz and above the bitstream ending '_hf' must be used. This bitstream has been built with the High Frequency Camera Link de-serializer.

For the examples that follow, one of the standard bit-streams supplied on the CD will be downloaded to your module, depending on your module type. When using one of these standard bit-streams, it will be necessary to understand what that bit-stream is doing.

Functional Block Diagram



Area-scan Camera
Conn.A
Camera Link Area-scan Signal Interface
Automatic Region of Interest Detection
RS-232 Interface
Capture Region of Interest
Frame Control
HSB Message Interface
HERON Interface
Output Packing and Synchronisation
511x32 CoreGen FIFO
HERON FIFO Write Interface

Camera Inputs

The Camera Link interface transmits camera control and data signals using the LVDS signalling standard. In order to interface to LVDS signals the correct input and output buffer components must be used in the FPGA design. This is done inside the file ‘top.vhd’. The module expects as inputs one LVDS camera clock and four LVDS camera data lines.

For the HERON-FPGA12 Camera Link example, no outputs are driven by the FPGA as the Virtex-4 FX does not support 3.3V LVDS signalling, only 2.5V.

The example bit-streams require the camera signals to be provided on Connector A of the HERON-FPGA12. The connections that are required are shown in the table below.

Connector A	Camera Link Signal
CONN_A0	X0+
CONN_A1	X0–
CONN_A2	X1+
CONN_A3	X1–
CONN_A4	X2+
CONN_A5	X2–
CONN_A6	X3+
CONN_A7	X3–
CONN_A10	Xclk+
CONN_A11	Xclk–

Signal Voltage Levels

The Camera Link standard uses Low Voltage Differential Signalling (LVDS) for the electrical interface layer. The LVDS signalling standard is as the name suggests, differential. Therefore, one LVDS connection will involve a positive signal and a negative signal complement. In order to correctly connect to LVDS a Xilinx FPGA must be used that supports the LVDS standard.

HERON-FPGA modules are available with either Spartan-II, Virtex-II or Virtex-4 family devices fitted. The Spartan-II family does not support LVDS signalling however, while the Virtex-II and Virtex-4 do. In the case of the Virtex-4 device, LVDS outputs are only supported at 2.5V and not 3.3V. As Camera Link requires 3.3V LVDS signalling the Virtex-4 FX FPGA can only be used with LVDS inputs.

In the case of the Virtex-II (and Virtex-II Pro) both 3.3V LVDS inputs and 3.3V LVDS outputs are possible so the Virtex-II based modules may be used with the Camera Link example.

Please note: a 100-Ohm termination is required for every LVDS signal pair that is received by the Virtex-4 FX FPGA. However in the case of the Camera Link example this is automatically implemented for you by the use of DCI termination resistance which is specified as part of the design source.

Camera Speeds

The bit-streams supplied by HUNT ENGINEERING have been specified for operation using a Camera pixel clock frequency of 50MHz or less. There are two versions of the bitstream, one built using a Low Frequency de-serializer for pixel clocks up to 24MHz and one built using a High Frequency de-serializer for pixel clocks above 24MHz.

The Camera Link standard uses 'Channel Link' for the physical layer. Channel Link consists of a driver and receiver pair. The driver accepts 28 bits of data and a clock. The data is serialised 7:1 into four data streams and transmitted with a dedicated clock over 5 LVDS pairs. The receiver accepts the four LVDS data streams and clock and then recreates the original 28-bit data stream along with the clock.

For those users who have cameras with pixel clocks above 50MHz the User Constraints File for the project needs to be modified to reflect the particular operating frequency. When this has been done the bitstream should be regenerated and the place and route report file checked to see those time-specifications were met.

Automatic Region of Interest Detection

The module AUTO_ROI performs automatic region of interest detection. The module allows the frame size and active areas of the image to be automatically detected and read over the HSB message interface.

This information, if used, enables the controlling software to automatically calculate the window for the capture region of interest process by using values that directly relate to the observed operation of the camera.

Capture Region of Interest

The module ROI performs a region of interest operation by using a pixel counter and a line counter, along with a pixel start position, pixel stop position, line start position and line stop position.

The pixel start, pixel stop, line start and line stop values are all programmed over the HSB message interface. Each of these values can be pre-calculated using the Automatic Region of Interest logic, or they can be set directly from values supplied by the user.

Frame Control

After the capture region of interest has processed each frame, whether that frame will be output or not depends on the frame-control component. The continuous mode and N-frames mode are programmed over the HSB message interface.

Output Packing and Synchronisation

One word, equal to 0 is added at the end of each camera line, apart from the last line of a frame, where one word with all bits set high (the value FFFFFFFFh) is added to indicate end of frame. In order to avoid the synchronisation values being present in the data, the bottom byte of camera data is modified to prevent the values 00h and FFh (a value of 00h will become 01h and a value of FFh will become FEh).

CoreGen FIFO and HERON FIFO Interface

After the data has been processed by the output packing and synchronisation component, the data is fed into a Core-generator generated FIFO. The FIFO is a 32-bit word, 511 word deep asynchronous FIFO built using Block RAM.

When there is any data in the CoreGen FIFO, it is read out and placed in the HERON FIFO Write Interface. The FIFO number used for output is programmed over the HSB message interface. If the HERON FIFO becomes full, eventually the CoreGen FIFO will become full and camera data will be lost. If the CoreGen FIFO becomes full, LED0 will become illuminated.

RS-232 Interface

Camera Link provides two LVDS pairs for serial communication between the ‘frame grabber’ and camera. The first LVDS pair ‘SerTC+/SerTC–’ is provided for communication To the Camera (TC) and the second pair ‘SerTFG+/SerTFG–’ is provided for communication To the Frame-Grabber (TFG).

With the HERON-FPGA12 Camera Link example, the RS-232 functionality is not available as it requires 3.3V LVDS outputs which is not supported by the Virtex-4 FX device.

HSB Control Registers

The following table defines the control registers that can be set by sending messages to the FPGA using HSB.

HSB Address Byte (decimal)	Register Function	Description
0	Pixel Start 0 Register	LSB of pixel start register for Capture Region Of Interest (register is 12bits)
1	Pixel Start 1 Register	Top 4 bits of pixel start register for Capture Region Of Interest (register is 12bits)
2	Pixel Stop 0 Register	LSB of pixel stop register for Capture Region Of Interest (register is 12bits)
3	Pixel Stop 1 Register	Top 4 bits of pixel stop register for Capture Region Of Interest (register is 12bits)
4	Line Start 0 Register	LSB of line start register for Capture Region Of Interest (register is 12bits)
5	Line Start 1 Register	Top 4 bits of line start register for Capture Region Of Interest (register is 12bits)
6	Line Stop 0 Register	LSB of line stop register for Capture Region Of Interest (register is 12bits)
7	Line Stop 1 Register	Top 4 bits of line stop register for Capture Region Of Interest (register is 12bits)
8	Frame Control Register	Set number of frames to capture and send (range 1 to 15, value set in bottom 4-bits) Or, set bit 4 for continuous capture (i.e. write 0x10)
9	Camera Control Register	Bit 0 is used to directly set Camera Link Camera Control 1 (CC1). Bit 1 is used to directly set Camera Link Camera Control 2 (CC2). Bit 2 is used to directly set Camera Link Camera Control 3 (CC3). Bit 3 is used to directly set Camera Link Camera Control 4 (CC4). Bit 4 is used to control the ODD-ONLY function. When this bit is set high only odd pixels are output in each line, and only odd lines are output in each frame. Bit 5 is used to set the polarity of the Camera Link control signals, FVAL, LVAL and DVAL. Set to 0 if all signals are active low. Set to 1 if all signals are active high.

HSB Address Byte (decimal)	Register Function	Description
9	Camera Control Register	Bit 6 is used to indicate whether the connected camera provides a Camera Link DVAL signal. Set to 0 if no DVAL signal is provided. Set to 1 if a DVAL signal is provided. Bit 7 is used to reset the Auto-Region-of-Interest logic. Set to 1 to assert reset and then set to 0 to de-assert reset.
10	FIFO Number Register	One Hot setting for the HERON Output FIFO number that the data will be sent on.
11	Camera Mode Register	Bits 0 to 3 MUST always be written with the value 4. Bits 4 and 5 are used to control a data shift which is required when using 10 or 12 bit Camera Link cameras, as follows: Bit 5 / Bit4 / Function / Data Shift 0 / 0 / 8-bit camera / none 0 / 1 / 10-bit camera / right x 2 1 / 0 / 12-bit camera / right x 4 1 / 1 / 14-bit camera / right x 6

The following table defines the status registers that can be read from the FPGA using HSB.

HSB Address Byte (decimal)	Register Function	Description
0	First Pixel 0 Register	Returns the bottom byte of the 'First-Pixel' value generated by the Auto-Region-of-Interest logic.
1	First Pixel 1 Register	Bits 0 to 3 return the top 4-bits of the 'First-Pixel' value generated by the Auto-Region-of-Interest logic. Bits 4 to 6 are undefined. Bit 7 returns the state of the Auto-Region-of-Interest Logic. 0 indicates that the Auto-ROI values cannot be used. 1 indicates the Auto-ROI values are valid and safe to use.
2	Last Pixel 0 Register	Returns the bottom byte of the 'Last-Pixel' value generated by the Auto-Region-of-Interest logic.
3	Last Pixel 1 Register	Bits 0 to 3 return the top 4-bits of the 'Last-Pixel' value generated by the Auto-Region-of-Interest logic. Bits 4 to 7 are undefined.
4	First Line 0 Register	Returns the bottom byte of the 'First-Line' value generated by the Auto-Region-of-Interest logic.
5	First Line 1 Register	Bits 0 to 3 return the top 4-bits of the 'First-Line' value generated by the Auto-Region-of-Interest logic. Bits 4 to 7 indicate bit activity in the 24-bit 'Base Configuration' Camera Link data (from bit 23 down to bit 0), as follows: Bit set high — Indicates 4 — Activity in bit 9 5 — Activity in bit 11 6 — Activity in bit 13 7 — Activity in bit 15
6	Last Line 0 Register	Returns the bottom byte of the 'Last-Line' value generated by the Auto-Region-of-Interest logic.
7	Last Line 1 Register	Bits 0 to 3 return the top 4-bits of the 'Last-Line' value generated by the Auto-Region-of-Interest logic. Bits 4 to 7 are undefined.

HSB Address Byte (decimal)	Register Function	Description
14	Receive Data Register	A read from this register returns the next data byte available in the RS-232 receive buffer.
15	Receive Status Register	A read from this register returns the number of data bytes in the RS-232 receive buffer.
16	Pixel Total 0 Register	Returns the bottom byte of the ‘Pixel-Total’ value generated by the Auto-Region-of-Interest logic.
17	Pixel Total 1 Register	Bits 0 to 3 return the top 4-bits of the ‘Pixel-Total’ value generated by the Auto-Region-of-Interest logic. Bits 4 to 7 return zero.
18	Line Total 0 Register	Returns the bottom byte of the ‘Line-Total’ value generated by the Auto-Region-of-Interest logic.
19	Line Total 1 Register	Bits 0 to 3 return the top 4-bits of the ‘Line-Total’ value generated by the Auto-Region-of-Interest logic. Bits 4 to 7 return zero.

Running the Tutorial

This tutorial covers downloading a Camera Link area-scan camera interface into the application FPGA of a HERON-FPGA12, followed by building and running the example DSP program to capture and store an image captured by the camera interface. To run the tutorial, follow this sequence:

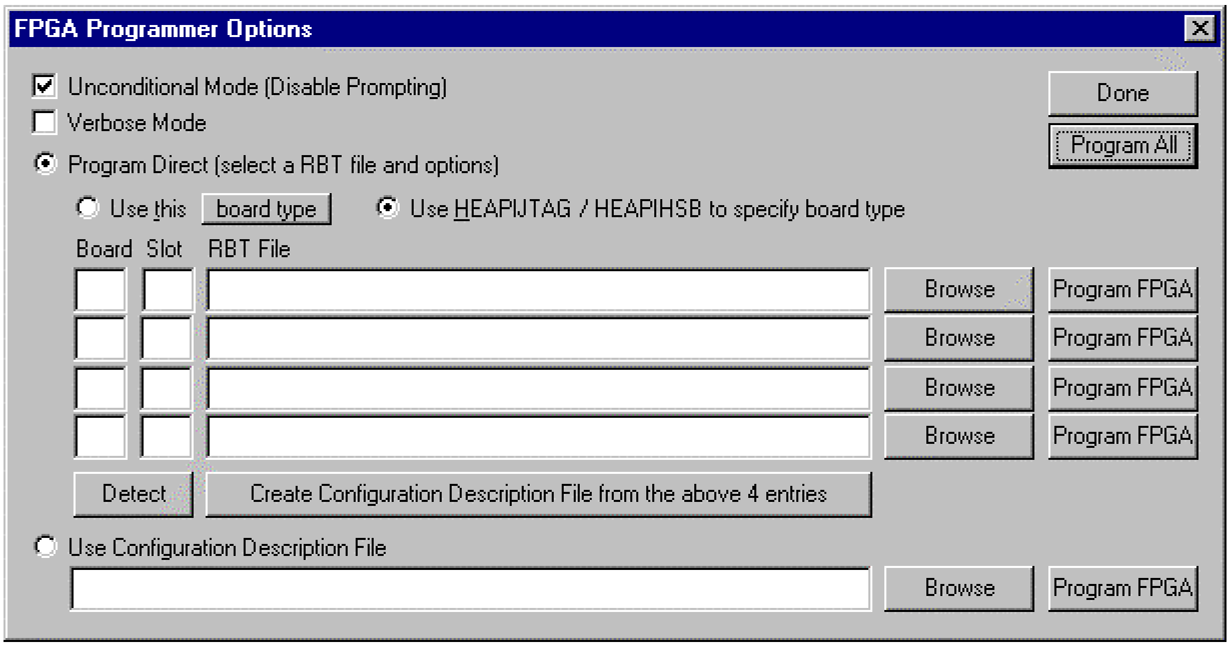
1) Select "Getting Started" from the Hunt Engineering CD menu, then "to start using FPGA modules and tools….", then "Examples & IP by Function" and finally select "Camera Link Area-scan Camera Interface (FPGA12)". This should be how you reached this document.

   On the same screen, click on the word "Files" to the right of "Camera Link Area-scan Camera Interface". Windows Explorer will open in the \fpga\fpga12v1 directory of the CD. Here you will see a directory tree below the directory \CamLink_Cam. These are the files for the Camera Link area-scan camera example.

   There is also a zip file, which is a zip of the entire tree for this module type. This is provided because restoring files from this zip file will restore files that are not set to read only. If you simply copy the files from the CD you will need to set the files in the Camera example directory so that they do not have the read only attribute set.

2) The next step is to program the FPGA by downloading the Camera example bit-stream. You will need to do this whenever the FPGA program has been changed, or when the PC has been powered off.

   Select START → PROGRAMS → HUNT ENGINEERING → Program HERON-FPGA You should see:



[✓] Unconditional Mode (Disable Prompting)
[ ] Verbose Mode
[✓ Program Direct (select a RBT file and options)]
Use this board type [✓ Use HEAPIJTAG / HEAPIHSB to specify board type]
Use Configuration Description File

   Use the Detect button to find your FPGA module, and the Board and Slot should get filled in for you. Then use the Browse button to select the bit-stream to use. You need to go to the directory of the CD that you were just in, and enter the 'Camera' directory. Here there will be several '.rbt' files, which are the bit-stream files. The name shows the number of gates and the package, e.g. 4vfx12ff668.rbt is for a Virtex-4 FX FPGA in a FF668 package.

   Be certain to use the appropriate file according to the pixel clock frequency of your camera.

   Now select "Program FPGA" or "Program All" to download the bit-stream. The configuration you have set up will be remembered for the next time so you do not have to go through the browsing process every time.

Actually the program that does the downloading is an executable that will have been installed in your %HEAPI_DIR%\utils directory and is called hrn_fpga.exe. This can be called directly from your own program, or perhaps from your autoexec.bat file. The program takes all of the options as command line parameters so that you will not have to answer prompts. Use hrn_fpga -h to see the options.

The FPGA download program is also invoked if you are using the Server/Loader plug-in and have not ticked the tick box 'Skip FPGA'. The bit-stream file used is defined in the network file being used.

A useful indicator is the "DONE" LED fitted to HERON modules that have an FPGA. It will be on after power-up or during a reconfiguration, and will only go out if the FPGA is programmed correctly. Check this to ensure the FPGA is programming correctly.

3) Now you need to understand how your system is set up. To run the example you will need to have a DSP module in your system. The example defaults to a DSP module in slot 1 and a HERON-FPGA12 in slot 2. Other slot positions may be used but if you do so, you will need to modify the example.c source file and, in the case of the HEPC9, the Server/Loader network file as well.

The DSP Example

Make a directory for the DSP example on your hard drive. In your Explorer window (from stage1) change to the \fpga\fpga12v1\CamLink_Cam\dsp directory. Copy the all of the files from there into the directory that you just made.

Now open Code Composer Studio and select the "Create new HERON-API Project" plug-in to generate a new project for your DSP module. The project needs to be made around the source file, example.c that was copied as described above. If you are not sure how to do this, then refer to the "Getting started with C6000" section of the CD and the tutorials there.

After the project has been created, you will need to add the source file bmp.c to the project.

With this done the project should build without any other changes, unless you wish to test an RS232 connection between your camera and the HERON-FPGA module. If you wish to test RS232 you will need to edit the header file 'rs232.h' according to your camera type.

Please note, the example program has been written to capture a single frame using a region of interest automatically created from the values output by the Automatic Region-of-Interest logic, whatever your camera type.

What this means is that you should not need to modify any source code before building the example. However, if when running the example you fail to get a picture then you should consult the section at the end of this document that discusses camera differences and how the Automatic Region of Interest logic functions.

Now you need to understand which FIFO number the DSP will use to access the FPGA, and which FIFO number the FPGA will use to access the DSP.

HEPC9

For an HEPC9, you can use the heartconf utility to make the connections you want, or you can use the default routing jumpers to make a simple connection. It is also possible to use the Server/Loader plug-in to configure HEART via the network file it uses.

For this example, the Server/Loader network file will be used, but if you are interested in the other methods, then for full instructions on using the heartconf utility, or setting the jumpers please refer to the documentation for the HEPC9.

The example assumes that you have a HERON module in slot 1 and a HERON-FPGA9 in slot 2. The network file provided connects FIFO 2 of slot 1 to FIFO 3 of slot 2. This works with the settings used in the example.c source file.

In order to use a different combination of FIFOs, then the DSP_FIFONO and FPGA_FIFONO #defines must be changed in the file example.c. Also the HEART configuration must be changed in the network file.

When using a HEPC9, you could connect any FIFO you choose from the DSP module to the FPGA module. In this example, the default values of FIFO 2 and FIFO 3 allow the same software to work on a HEPC8, with the same slot assignments.

The following lines are taken from the network file provided.

```
# Using API
BD API HEP9A 0 0
#-----------------------------------------------------------
# Nodes description
# ND  BD_nb  ND_NAME  ND_Type  CC-id HERON-ID  filename(s)
#-----------------------------------------------------------
  ibc   0     ibc1    normal         0x06
  pcif  0     host1   normal         0x05
  c6    0     mod1     root          0x01   example.out
  fpga  0     mod2    normal         0x02   2vp7ff672.rbt
#-----------------------------------------------------------
#       from:slot  fifo  to:slot   fifo   timeslot
#-----------------------------------------------------------
heart     host1     0      mod1     0        1
heart     mod1      0      host1    0        1
heart     mod1      2      mod2     3        1
heart     mod2      3      mod1     2        1
```

For the network file to be correctly used to configure the FIFO connections, you need to ensure that the Server/Loader plug-in has been correctly set up. Once the project has been successfully built, start the Server-Loader plug-in.

In the plug-in, click on the Browse button and navigate to the directory to which the example files were copied. Select the file named network and click open.

With the bit-stream downloaded to the FPGA, and the DSP example correctly configured and built, the example program can be run. Before you do so, ensure that the camera is correctly powered up and that the correct signal connections have been made between the camera and Connector A of the HERON-FPGA9.

Also ensure that the lens-cap has been removed from the camera and that the lens iris is at least partially open. This is required to ensure that the Automatic Region of Interest functions correctly as it calculates image areas by detecting active parts of the image.

The program needs to be run using the HUNT ENGINEERING Server/Loader plug-in. The program will be run by clicking on the Start S/L button. If there is no tick in the ‘Skip FPGA’ box, then the FPGA of the HERON-FPGA9 will be re-configured according to the .rbt file specified in the network file. For this example, we have already configured the FPGA (this can be checked by looking at the DONE LED of the HERON-FPGA9 to see that it is off), so it will be better at this point to tick the box ‘Skip FPGA’.

When the program runs, it will reset the Automatic Region-of-Interest logic and begin to detect the frame size and active area of the image. When this information has been obtained it will then send all of the set-up information required over HSB. This information includes setting the pixel-start, pixel-stop, line-start and line-stop values required by the capture region-of-interest, as well as setting the mode of operation to 4x8, and programming the HERON Output FIFO number according to the value of the FPGA_FIFONO #define in example.c.

When all of the HSB configuration messages have been sent, the program will request one frame of data by sending the value 1 to the FRAMES input of the FRAME_CONTROL component (again over HSB).

This frame will then be received by the DSP on the FIFO set by the DSP_FIFONO #define. The pointer of the buffer the frame was stored to is then passed to the bitmap creation functions and a bitmap file will be made. The name of the bitmap file will be ‘image.bmp’. When the program has completed and the Server/Loader indicates program execution has finished, this file can be viewed to see the image that was captured.

Assuming that an image has been successfully captured, you may continue either with developing your DSP application to further process the image, or you may modify the FPGA program to perform image processing inside the FPGA.

If you have been unsuccessful in capturing an image please refer to the last section of this document, which discusses common errors and how to resolve them.

The following sections of this document describe the VHDL components that are provided in the example, as well as explaining what you will need to consider in order to modify the FPGA program.

The FPGA Program

At this stage you should have been able to run the Camera Link Area-scan Camera example on the FPGA and DSP and should have captured and viewed an image. If an image was successfully captured, then at this stage you will probably need to further develop the FPGA program. The following assumes that you are using the Xilinx ISE tool-set.

This part of the tutorial will show you how you can make some very simple modifications to the FPGA program.

The example projects for ISE are shipped on the HUNT ENGINEERING CD. Using these projects will allow you to run the complete design flow, from RTL-VHDL source files to the proper bit-stream, 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.

You may also visit ALSE's Web site where you will find design tips, useful links, design examples, etc… at : http://www.alse-fr.com (then click on the "English version" banner).

Preparing ISE

Make sure ISE is properly installed with the XST-VHDL flow and ad-hoc device support.

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-FPGA9 the correct directory is “fpga12v1”.

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

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 “CamLink_Cam” directory 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. Please note, the Camera Link example uses a separate version of the “Common” directory. This is because the file “top.vhd” has been modified to include LVDS IO buffers for the connection to the Camera Link camera.

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 Camera Project

In the tree that you have just copied from the CD, open the `CamLink_Cam` sub-directory. You should see some further sub-directories.

* `Common` holds the common project files, including a modified version of Top.vhd.

* `CamLink_Ex` holds the Camera Link example

Below the `CamLink_Ex` directory you should see the following sub-directories.

* `ISE` holds the ISE project files

* `Src` holds the application-specific source files

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

You need to select the correct file with the .npl extension from the `ISE` directory under the `CamLink_Ex` directory.

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

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

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

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

Project's Functional Parameters

Double click on "`user_ap`" 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 edit any file other 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 user_ap1.vhd file there are the settings that you can use to affect your design (in this case the camera example). The idea is that settings that are often changed are found here.

1. 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.

In that case you provide a frequency for that clock on the signal SRC_FCLK_G. For module carriers like the HEPC9, HECPCI9 and HERON-BASE2 this clock must be greater than or equal to 60Mhz and less than or equal to 100Mhz. If you have another carrier card please check it's user documentation for the limits of it's FIFO clock frequency.

The signal FCLK_G must then be used in your FPGA design as the correct phase of this clock to sample the FIFO data with.

If you have different input and output FIFO clocks then select FCLK_G_DOMAIN as False and then you use the individual SRC_FCLK signals to drive the frequencies and the FCLK signals to clock your logic.

You also need to consider the timing constraints that are defined in the .ucf file for your design. Actually if you use a time specification that is more strict than needed there is no problem, so the standard .ucf file have the clocks specified at 100Mhz. If the project builds (as example1 does) with this specification it is still guaranteed to work at lower clock speeds. If you add new clock nets into your design then you need to add new timing constraints into your design.

Creating the Bitstream for the Camera Example

Once the project has been opened as described above:

1. In the "Sources in project" window (Project Navigator), highlight (single-click on) "top".

   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 of 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 bit-stream:
   Note that this stage runs a Design-Rule-Check (DRC). The DRC can potentially detect anomalies created by the Place-and-Route phase.

   When the processing ends, the proper bit-stream file, with extension `".rbt"` can be found in the project directory.

4. In the "Pin Report" verify a few pins from the busses, to check that the ucf file has been used. If you see different assignments, STOP HERE, and verify the UCF file selected for the project.

You can now download this file on your FPGA board and see how it works.

Note that the user_ap level includes a very large counter which 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.

If the LED does not flash, try a hardware reset. The DLL used in the clock generation often does not start until a hardware reset. If the LED still does not flash we recommend that you shut down the PC or reprogram the device using a "safe" bit-stream. 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, re-run the camera DSP example, then return to the original camera example.

The FPGA Design

Camera Inputs

The first major functional block in the Camera example is the VHDL module CLINK. This module is used to directly interface to the Camera Link interface of the camera that you are using. This module expects as inputs, four 7:1 serialised camera data signals, and a camera clock. From the four serialised data streams, the module recovers the original 28-bits of camera-link camera data.

From the 28-bits of data, 24-bits of camera image data are obtained, along with a data valid, line valid and frame valid control signals. The 24-bits of camera data are presented in the bottom 24-bits of the 32-bit data output by the CLINK module (data_out (31 downto 0)). The top byte is set to zero.

The control signals are used to generate a data-valid signal (dvalid_out), end of line control signal (eol_out) and end of frame control signal (eof_out), along with the pixel clock output (pixel_clock).

The CLINK module provides the capability to perform a byte shift so that the 8-bits of camera data processed by the FPGA match the top 8 active bits from the Camera Link camera. For an 8-bit camera, this byte shift is not required. However, when using 10-bit or 12-bit cameras, a byte shift right of 2 or 4 bits respectively is required to ensure that only the top 8-bits of each pixel data are processed.

Signal Voltage Levels

The Camera Link standard uses Low Voltage Differential Signalling (LVDS) for the electrical interface layer. The LVDS signalling standard is as the name suggests, differential. Therefore, one LVDS connection will involve a positive signal and a negative signal complement. In order to correctly connect to LVDS a Xilinx FPGA must be used that supports the LVDS standard.

HERON-FPGA modules are available with either Spartan-II, Virtex-II or Virtex-4 family devices fitted. The Spartan-II family does not support LVDS signalling however, while the Virtex-II and Virtex-4 do. In the case of the Virtex-4 device, LVDS outputs are only supported at 2.5V and not 3.3V. As Camera Link requires 3.3V LVDS signalling the Virtex-4 FX FPGA can only be used with LVDS inputs.

In the case of the Virtex-II (and Virtex-II Pro) both 3.3V LVDS inputs and 3.3V LVDS outputs are possible so the Virtex-II based modules may be used with the Camera Link example.

Please note: a 100-Ohm termination is required for every LVDS signal pair that is received by the Virtex-4 FX FPGA. However in the case of the Camera Link example this is automatically implemented for you by the use of DCI termination resistance which is specified as part of the design source.

Camera Speeds

The .ucf file for the example project contains time-specs that cover the use of pixel clocks in two different frequency ranges. Each time the project is built only one set of time constraints must be selected, with the un-required group being commented. The group of constraints used will depend on whether you are using the low frequency de-serializer logic or high frequency de-serializer logic.

As long as the appropriate high or low frequency de-serializer is selected and the time-specs are met when the design is built any camera link clock frequency can be used.

Automatic Region of Interest Detection

Data output by the CLINK component is monitored by an automatic region of interest function. This process counts the number of pixels in a line and number of lines in a frame to create the Pixel-Total and Line-Total values.

The process also detects whether there are any dark (black) pixels on the left side, right side, top or bottom of the image, creating First-Pixel, Last-Pixel, First-Line and Last-Line values respectively.

These six values can be read using the HSB message interface. This allows controlling software to automatically detect where the active area of the image is. The values returned by the Automatic Region-of-Interest can be used in calculating the values with which to program the Capture Region of Interest.

The advantage of doing so over calculating the values by hand is that correct frame capture can be guaranteed. For example, if a camera is operating with 400 pixels in a line and 500 lines in a frame, and the region of interest and DSP read process (typically HeronRead) are expecting 512 pixels x 512 lines from hand calculated values, then the capture process will never complete. This is because more data is expected than can be generated in one frame by the camera.

Using the automatically calculated values, the frame capture should always complete, as the software will be working with a frame size that is achievable. This is because the frame size used has been calculated from monitoring what the camera produces.

Capture Region of Interest

Data output by the CLINK component is put through a programmable region of interest. The module ROI performs a region of interest operation by using a pixel counter and a line counter, along with a pixel start position, pixel stop position, line start position and line stop position.

Starting with the first pixel in each line of data, the pixel counter starts at 0 and counts up. If the current pixel count is equal to or greater than the pixel start value, AND if the pixel count is less than or equal to the pixel stop value then that pixel is output. All other pixels in the line will be discarded. Similarly, starting with the first line in each frame of data, the line counter starts at 0 and counts up. If the current line count is equal to or greater than the line start value, AND if the line count is less than or equal to the line stop value then that line is output. All other lines in the frame will be discarded.

In the example the pixel start, pixel stop, line start and line stop values are all programmed over the HSB message interface. The values used can be calculated from the values returned by the Automatic Region of Interest logic.

Frame Control

The component FRAME_CONTROL controls when frames are output over the HERON FIFO Write Interface. The FRAME_CONTROL component can operate in two ways. It can be set-up to continuously output frames by setting the ‘continuous’ input high, or can be set-up to output between N frames (where N is 1 to 15) by setting a value on the ‘frames’ bus input and asserting the load signal.

After the region of interest has processed each frame, whether that frame will be output or not depends on the frame-control component. In the example the continuous mode and N-frames mode are programmed over the HSB message interface.

Output Packing and Synchronisation

The component PACK_SYNC takes the frames that are output by the FRAME_CONTROL component and packs the data ready for transmission through the HERON FIFOs. How the data is packed will depend on the mode of camera operation.

The pixel stream that is processed by the CLINK, ROI and FRAME_CONTROL components is a 32-bit wide data stream. This data stream may contain valid data only in the bottom byte, or only in the bottom two bytes or in all four bytes.

For 4x8 mode, only the bottom data byte carries camera data. In this mode, each 32-bit word sent to the HERON FIFOs is packed with four bytes of data. The data in the bottom byte is the first pixel in time, and the data in the top byte is the last pixel in time. For 2x16 mode each 32-bit FIFO word is packed to contain two 16-bit pixels. The bottom half of the word will contain the first pixel in time. Finally, in 1x32, one pixel is output in one 32-bit FIFO word. The 4x8, 2x16 and 1x32 mode control inputs are configured over the HSB message interface.

As well as data packing according to mode, this component also adds synchronisation markers to the data. One word, equal to 0 is added at the end of each camera line, apart from the last line of a frame, where one word with all bits set high (the value FFFFFFFFh) is added to indicate end of frame. In order to avoid the synchronisation values being present in the data, the bottom byte of camera data is modified to prevent the values 00h and FFh (a value of 00h will become 01h and a value of FFh will become FEh).

CoreGen FIFO and HERON FIFO Interface

After the data has been processed by the output packing and synchronisation component, the data is fed into a Core-generator generated FIFO. The FIFO is a 32-bit word, 511 word deep asynchronous FIFO built using Block RAM.

When there is any data in the CoreGen FIFO, it is read out and placed in the HERON FIFO Write Interface. The FIFO number used for output is programmed over the HSB message interface. If the HERON FIFO becomes full, eventually the CoreGen FIFO will become full and camera data will be lost. If the CoreGen FIFO becomes full, LED0 will become illuminated.

FPGA Resources Used

It may be interesting for someone who wants to develop their own FPGA design based on the example, to judge how much of the FPGA is used by the camera interface logic. The standard IP uses 20% of the logic in the xc4vfx12 Virtex-4 FX FPGA.

There are also block rams used by the CoreGen FIFO, but a newly developed design would replace or re-use that FIFO.

Developing the Camera Link Example

The Camera Link area-scan example is organised as a sequence of inter-connecting function blocks that each perform a separate process on a stream of camera data.

The functional blocks in the example provided include a block to interface directly to the camera signals, an automatic region of interest detection block, a block to reduce the image through a 'Region-of-Interest', a block to control when frames of data are output, and a block to pack and format the data ready for the next stage of processing.

Each of these blocks could be easily replaced, or removed to perform a different set of processing on the camera data.

For example, you may wish to perform no region of interest, and simply pass the entire camera frame from the camera directly to the DSP or other processing. In this case, you could delete the `ROI` component instantiation, and connect the output of the `CLINK` component to the input of the `FRAME_CONTROL` component. This would be possible because the pixel-data-and-control interface of these three components has been designed to be the same.

When developing a new FPGA program for the HERON-FPGA12, by making a new component that interfaces to camera data in the same way, you will be able to easily insert that component into the processing chain that currently exists in the area-scan example.

Using the Pipelined-Pixel-Stream Format

The components `ROI` and `FRAME_CONTROL` of the area-scan example process data in a format that we will call the pipelined-pixel-stream format. As well as these two components, the `CLINK` component is used to take real-world camera signals and create an output stream that is in the pipelined-pixel-stream format, and the `PACK_SYNC` component takes data in that format and outputs as 32-bit words to a HERON FIFO.

This format is not a proprietary format, and can be changed in your design if necessary. It has simply been created so that a consistent starting point exists for fitting together separate functional units in an imaging application.

At any point in a pipeline that is using the 'pipelined-pixel-stream' format, there must be four control signals and N-bits of data. The control signals are `PIXEL_STROBE`, `DVALID`, `END_OF_LINE` and `END_OF_FRAME`. For the example provided, N is set to 32.

The `PIXEL_STROBE` is the pixel strobe or clock signal that must be used to clock all synchronous elements in the pixel processing pipeline. This signal will be typically derived from a clock signal output by the camera.

The `DVALID` signal is a 'data valid' signal. When asserted (set high) it must indicate that on a rising edge of the `PIXEL_STROBE`, there is valid data on the 32-bit data bus.

The `END_OF_LINE` and `END_OF_FRAME` signals are used to 'frame' the camera data. After the transmission of the last pixel in a line, the `END_OF_LINE` signal must be asserted (set high) for one pixel-clock period. Similarly, after the transmission of the last pixel in the last line, the `END_OF_FRAME` signal must be asserted (set high) for one pixel-clock period.

The signals `DVALID`, `END_OF_LINE` and `END_OF_FRAME` must be asserted in such a way that only one of them is high on any clock edge.

Changing the CLINK Component

The `CLINK` component performs an important task in the Camera Link example, in that it takes the signals that are provided by the Camera Link interface, and fits them to the format of the pipelined-pixel-stream used by all of the other components. If you are working with a camera format that is not supported by the `CLINK` component provided, you will need to change or replace this component.

The component will typically need to register the signals on the connectors in IOB registers inside the FPGA to meet the timing requirements of the camera output interface. It will then need to generate the `PIXEL_STROBE`, `DVALID`, `END_OF_LINE` and `END_OF_FRAME` signals as well as N data bits, where N will depend on the width of the camera data. The component must ensure that the timing of the `DVALID` signal directly matches the timing of the data that is output.

Changing the AUTO_ROI Component

The Automatic Region-of-Interest component provided implements logic that monitors how many pixels are output by the camera in each line, and how many lines are output in each frame. The component also automatically detects the left hand edge, right hand edge, top edge and bottom edge of the image so that dark pixels can be removed from the image by the capture region of interest component.

As this component only monitors the activity of the data pipeline, and does not directly modify signals that feed into downstream components, it can be modified or removed from the design without effecting the image processing.

Changing the ROI Component

The Region-of-Interest component provided implements a region of interest by using a start and stop position in a line, and start and stop position for lines in a frame. Essentially, this component will reduce, in each frame, the number of clock cycles where the `DVALID` signal is asserted. For incoming pixels within the region-of-interest, the `DVALID` output will be high when the `DVALID` input is also high. However, for a pixel that falls outside the region of interest, the `DVALID` output will always be low.

The `END_OF_LINE` and `END_OF_FRAME` signals must be preserved by the region of interest so that any processing units downstream will still be able to reconstruct the image.

Changing the FRAME_CONTROL Component

The frame-control component uses the `END_OF_FRAME` signal to decide when a new frame is being received. It combines the information provided by this signal with logic that defines which frames to output. While a frame is to be output the signals `DVALID`, `END_OF_LINE` and `END_OF_FRAME`, must be passed through un-altered. For any frame not to be transferred all control signals must be de-asserted for the whole of that frame.

Changing the PACK_SYNC Component

The output packing and synchronisation component receives data in the pipelined-pixel-stream format, by outputs data in a format that is defined inside the component. In the area-scan example provided, this format is made to suit transmission of an image to a DSP. Data is packed into words, and synchronisation markers are added to enable receiving software to reconstruct the image.

This component can be made to do whatever you want, as the contents of the component will be decided by the format of the data that you wish to output.

Adding New Components

New components can be added to do you want, and can be made to interface to the camera data in any way you choose. However, if you want to add new components and want to use the pipelined-pixel-stream format, then you must consider the following points.

- The `PIXEL_STROBE` signal must be used to clock all synchronous elements that interface to the camera data and control signals `DVALID`, `END_OF_LINE` and `END_OF_FRAME`.
- All valid pixels input to the component will be qualified by the input `DVALID` signal being asserted (set high). An output `DVALID` signal must be generated for the component. For each valid pixel output by the component, the output `DVALID` signal must be asserted.
- The input signals `END_OF_LINE` and `END_OF_FRAME` indicate where a line ends and where a frame ends respectively. These signals must be passed through unaltered, apart from the case where whole frames of data are to be removed from the data stream. In this case, the signals may be de-asserted throughout the frames that are removed.
- The data output by a component must be valid in the same clock cycle as the assertion of the `DVALID` signal to which that data corresponds.

What to check if you can’t get the image you expected

The example program has been written to capture a single camera frame from many different types of Camera Link camera. The image when captured is written to a Windows Bitmap file so that it can be viewed in common image editing programs such as Microsoft Paint.

It may be possible that when you ran the example you were unable to capture an image, or that an image was captured but it was entirely dark, or that the image was smaller than expected. In either case, please read through the rest of this section, which is intended to help you find the cause of the problem.

Problem: Program did not capture an image and no Bitmap file was created

If the example program did not complete and no bitmap file was created, then this typically indicates that the DSP did not receive one complete frame of data from the FPGA module. There are a few reasons why this may have happened:

1. The camera you are using is not powered up at all, or is not powered up correctly. Please check the power supply that the camera is using and if necessary refer to the User Manual for your camera. In some cases, the camera has an LED on the back of the camera that when illuminated indicates the camera is correctly powered.
2. The camera is not correctly connected to the FPGA module. Please check that you have correctly connected the signals of the camera to the appropriate signals on Connector A of the HERON-FPGA12. The Connector signal assignment information at the beginning of this document should be used when checking the camera signal connections.
3. The camera is operating with a set of control signals different to what is expected by the HERON-FPGA12. Please check the User Manual for the camera type you are using. The example program supplied on the CD has been configured for a typical Camera Link camera. As such, it expects that the Camera Link Camera is providing a FVAL frame-valid signal that is active high (set to 1 when outputting valid frames) and a LVAL line-valid signal that is active high (set to 1 when outputting valid lines). If your documentation indicates that the camera is outputting active low control signals, please edit the ‘def.h’ header file of the example to set the `CAM_POLARITY` #define to 0 instead of 1.

Problem: An Image was captured but was entirely dark or smaller than expected

If the example program completed and created a bitmap file that was entirely dark or smaller than expected then this typically indicates some fairly simple errors that can quickly be fixed:

1. The image appears entirely black. Please check that the Lens cap has been removed.
2. The image appears very dark. Please open the Lens iris to let in more light.
3. The image is smaller than expected. This can be fixed in two ways. Firstly the example program is configured to use the automatic region of interest logic inside the FPGA. The automatic region of interest logic requires the camera to be generating images that are not very dark. In the case where the camera is generating dark images, the automatic region of interest may report a smaller frame size than is actually possible with a bright image. The solution is to open the lens iris to increase brightness of the image. Secondly, a different ROI mode can be selected in the ‘def.h’ header file of the example program. If ROI_MODE is set to 0, the example program when rebuilt will capture the whole frame output by the camera, including any black pixels surrounding the active image.