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

HUNT ENGINEERING

Imaging with FPGA Demo/Framework

An Example of capturing images with HERON-FPGA modules with standard IP, image processing on the FPGA using the Hunt Engineering Imaging VHDL and displaying the results on the host PC's graphics display. Originally written for Windows 98, but should work on any 32-bit Windows system.

Software Version 1.0
Document Rev B
P.Warnes 12/12/03

TABLE OF CONTENTS

Modern FPGAs can be used to build powerful image processing systems. Many standard imaging functions are easily programmed into an FPGA where their performance can easily outstrip the performance of more conventional processors.

HUNT ENGINEERING have produced some VHDL Image Processing Source Modules that perform many of these standard functions. This demo/framework is provided to a) demonstrate those VHDL modules and b) act as a starting point for customers that want to develop image processing applications using HERON systems.

There is an FPGA design for the HERON-FPGA5 that adds the imaging modules to the standard IP for a Camera Link camera. This is combined with a PC based windows program that controls that FPGA program and displays the results in a Windows window. Both of these items are provided as source code allowing them to be used as a starting point for your own development.

In addition there are bitstreams provided for other FPGA module types to allow the Imaging IP to be demonstrated on those modules and with both Camera Link and RS422 cameras.

In many imaging systems there is a requirement for display. Sometimes this is a "hard" requirement, where the display needs guaranteed update rates, or high resolution / high frame rates; however in many systems, the display is not crucial to the system's operation.

Examples of such systems include a display for focusing or aligning cameras, a confidence checking display, or a monitor to display processed results. In such cases, it is possible to use the host PC for display.

This is an example of using HUNT ENGINEERING imaging boards in this way. It captures video, performs some processing, then transfers the resulting video to the PC for display on the PC's monitor. This is done entirely using Windows function calls, so should be portable across all 32-bit Windows systems.

The example also demonstrates use of the HUNT ENGINEERING VHDL Image Processing modules to perform basic logical, arithmetic and filter operations. It can self-run, or be set to each task.

The software has been written using the HUNT ENGINEERING Host API software. This allows it to be easily ported from one platform to another.

Performance of the system depends very much on the bus bandwidth of the PC being used and the processor loading of that PC.

Notes:

1. The demo uses an image size of 384x384, and will work with any camera that has at least that number of pixels and lines. It is possible to re-compile to use a different image size, and it will automatically re-size the processing and display as long as the camera provides at least the image size that it has been compiled for.
2. The demo uses 8-bit monochrome video. The most efficient display settings for the application are 24-bit or 32-bit colour – this reduces colour space conversion to three copy operations rather than colour search / match operations. Frame rate may drop if you use 16-bit or 8-bit mode.

Overview

The demo as-is will work with a HUNT ENGINEERING system that is based on an HEPC9, with one of the supported FPGA modules in any slot. The demo will program the FPGA module with a special "frame-grabber and demo" bit stream. These bitstreams assume that the camera is connected as described in the corresponding standard frame-grabber IP.

The demo should run on any PC with any 32-bit Microsoft Windows installed on it; however, it will operate better on faster machines with accelerated graphics cards.

Camera's supported are Camera Link and RS422 area-scan camera's, with at least 384 x 384 visible pixels. The pixel size should be 14 bits or less. The demo detects if the camera is providing 8, 10, 12 or 14 pixels and takes only the top 8 bits for the demo.

The demo consists of a Windows host program, and bit-streams for the various FPGA modules. The bit-streams used are based upon the standard Camera Link IP and RS422 Camera IP on the HUNT ENGINEERING CD.

The same host program can be used with all of the combinations of camera type and FPGA module type by selecting the bitstream to be loaded onto the module. The names of the bitstreams supplied indicate the module type and camera type.

Use of APIs etc

The demo is based around HUNT ENGINEERING's standard APIs and utilities. These give it great portability, allowing us to switch easily from one hardware platform to another. We discuss them a lot in the following outline; here is an introduction to their function.

Host API is the standard software interface to any Hunt Engineering board from a host PC. It provides a simple mechanism to control the board and transfer blocks of data between the PC and the modules in the system. In this application it is used to pass images from the FPGA to the host.

Hardware Interface Layer This VHDL is provided for the FPGA modules to correctly interface to external hardware like the HERON FIFOs.

Imaging VHDL This is like a library of image processing primitives, written in VHDL. It is used to provide the image processing functions for the demo.

DSP Hardware

The demo has been tested on a variety of hardware, as follows:

- Various Camera-link digital cameras from Pulnix and Jai
- Pulnix TM 300 RS422 digital camera
- HERON-FPGA3,4 and 5 modules
- HEPC9 PCI carrier board

Full use of the APIs provided with HUNT ENGINEERING systems allows easy portability to other platforms; however bear in mind that other cameras may require different configuration, while different HERON-PFGA modules would require their own Hardware Interface Layer VHDL.

PC Hardware

The demo should be compatible with any 32-bit Windows system. Obviously, it uses considerable bus bandwidth for transferring video, and performs rapid bit-blits to place the video on the screen. Faster PC systems will perform this better, but the demo should still operate on older systems with no graphics acceleration and slower processors.

This raises a design issue we must be aware of – in many circumstances the DSP and framegrabber will be able to supply images far faster than the PC can display them. This is taken into account in the host software by checking for the sync positions in the data, and discarding incomplete frames.

Configuration File

There's a text file included in the demo that holds some parameters that are used by the demo to configure the camera used. This allows you to use the demo for some variety of Camera Link and RS422 cameras, and also gives you some degree of control over what the demo does.

The file is called 'camera.dat'. In here you can find the following parameters.

```
cam_control  = 0xf
```

This is a parameter that represents the camera control value.

```
cam_use_dval = 0
```

Different Camera Link and RS422 camera's may use slightly different protocols. One of them is whether the 'dval' signal is used or not. With this parameter you can change that setting.

```
cam_polarity = 1
```

Different Camera Link and RS422 camera's may use a different 'polarity'. In one case, a logic 1 may mean 'active', while for another camera a logic one may mean 'not-active'. With this parameter, you can indicate the polarity of the signals of the camera used.

FPGA Code

The FPGA code performs the following tasks:

- Interface to the Camera
- Detect some parameters about the image
- Apply some selection of the image information to process
- Process the images
- Send the processed data to the Host program for display.

The first elements of the FPGA design are directly taken from the standard camera interface IP provided on the HUNT ENGINEERING CD. The Auto-ROI, ROI, Frame

control etc is directly used from those IPs.

The Image processing Source modules also provided on the HUNT ENGINEERING CD are added to that standard IP, and the incoming image data applied to all of them at the same time. This allows the data for display on the PC to be selected by using a multiplexor that feeds the HERON FIFO data with the output of the selected Imaging function. There is a small FIFO implemented on the output of the multiplexor to help de-couple the incoming frames from the bursty transfers of the PCI bus.

The FPGA project supplied is for a HERON-FPGA5 module. This module has memory that is external to the FPGA, which allows some inter-frame processing to be made. For the purposes of the demo the way that this is used, is that there is a function to "capture" a frame into the SDRAM, which is then used as the "reference" for the inter-frame functions. The Host program has a check box where you can select that you have SDRAM on your module. This will enable the inter-frame modes in the demo.

The bitstreams for the module types that do not have external memory cannot include these functions, so they cannot be demonstrated in the demo. The SDRAM modes should not be selected for these modules.

Host Code

The host code is a Windows application written and compiled using Microsoft Visual C++. Understanding it requires some knowledge of the Windows operating system.

The application is split into three main sections:

1. User Interface / application code, which deals with the user interface, updating the display, moving the screen and so forth;
2. WndPROC, the window message handler. This function is registered as the handler for all messages for our display window.
3. FPGA threads, which load the FPGA modules, controls the output multiplexor from the FPGA and performs all data transfers from the board.

The application is multi-threaded. The main thread performs the User Interface code, while the second thread handles the DSP board. Splitting the application in this way greatly improves the usability of the system; user interface options (eg moving the window, selecting a menu, closing the application) can be performed while the application is reading the FPGA data.

Note WndPROC is called by the Windows operating system itself – there are no calls to this function in the code.

User Interface / Application Thread

This is the main thread of the application. It performs the following tasks:

1. Open a window for image display
2. Create two bitmaps and initialise with a test pattern
3. Start the FPGA code thread
4. Start the timer
5. Process Windows system messages

Once these tasks are performed, this thread simply polls the Windows message queue and despatches messages. In the main, these are either handled by the system or by the WndProc function.

Key messages arrive from the timer and from the user. Timer messages are used to switch the FPGA multiplexor to its next operating mode, while the user can request a switch manually.

WndProc

This function is called to handle all messages to our display window, from whatever source. Note that we don't call this function ourselves – it is registered as the handler for the window, and it is called by Windows directly.

There are three messages handled here that are worth noting:

WM_PAINT — This is the standard "Window Paint" message. Windows will send this message whenever the window needs an update – perhaps because it has been moved, or more of the window has been displayed; or part of the window has become valid.

This paints the window. Display is performed using a Windows BitBLT call – this copies the image directly to the screen. StretchBLT would perform scaling – a call to StretchBLT is in the code, commented out, should you wish to try this.

Note that to use StretchBLT you will need to change the window class – it is currently defined as non-resizable.

FPGA_IMAGE — This message is sent by the FPGA code thread. It indicates that a new image is available for display. The message causes the display window to be invalidated, causing a WM_PAINT event (See above).

FPGA_STATUS — This sets a flag for WM_PAINT, indicating that the status text has been changed. It also invalidates the screen. This causes WM_PAINT to redraw the status bar.

FPGA Thread

The host's interface code for accessing the FPGA is contained in a separate thread, started by the main application. This is not strictly necessary – it would have been possible to write the application with a single thread. However, multi-threading makes the code simpler and more robust for the user.

The code uses the HUNT ENGINEERING Hrn_fpga and Heartconf programs for starting the system. This performs all booting operations, such as resetting the board, loading the code and starting execution.

Once loaded, the host opens an API session for the board. This allows us to create our own data passing mechanism across the interface, allowing us to send commands to the board and receive images back.

Once booted, the code continuously reads images from the board. When a complete image is read, the FPGA thread checks for the correct alignment of the frmaesync, and if it is correct it sends message FPGA_IMAGE to the display window – this forces an update. The code also initiates a read of the next image.

If the Framesync is not at the correct location, this is because the Host has not responded

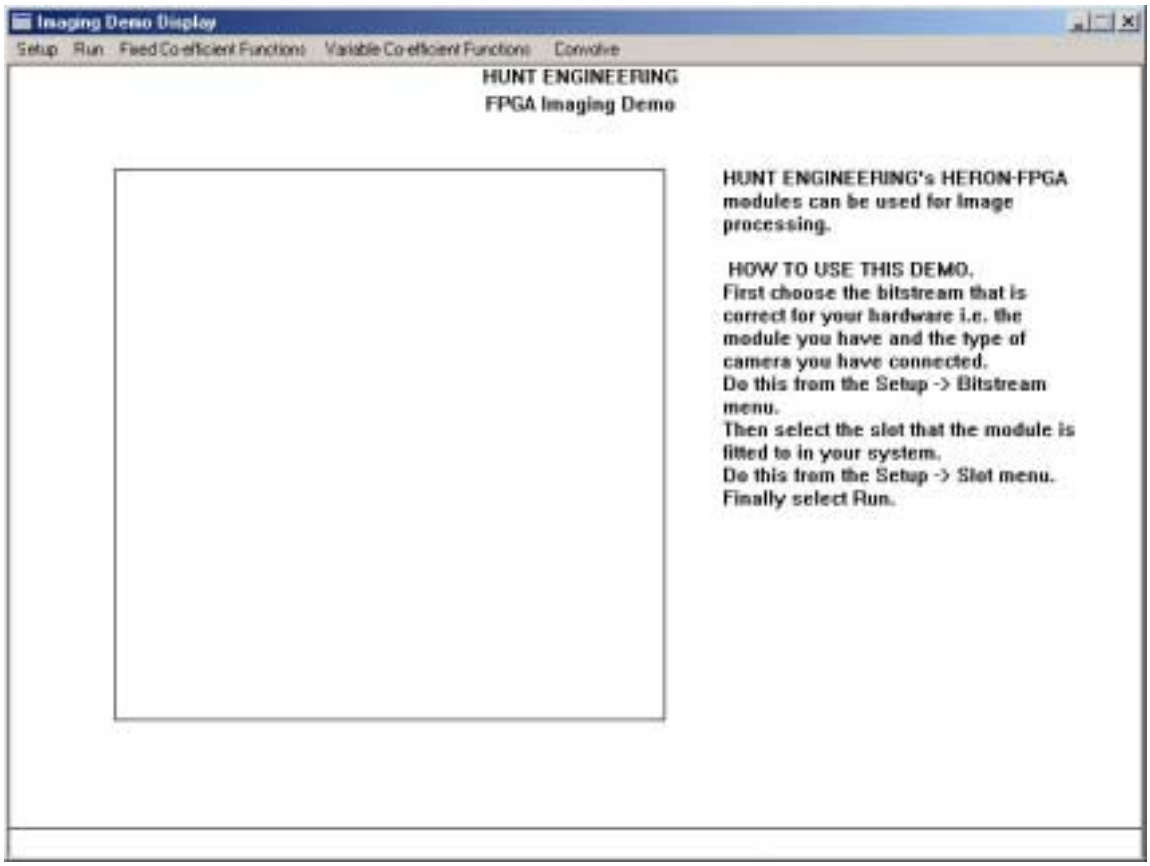
to the PCI interface quickly enough and the FIFO in the FPGA design has overflowed. Windows is not a real-time operating system and makes no claims to respond in a given time. In fact statements can be found to the effect that the interrupt response time can exceed $1/10^{th}$ of a second. To recover from this situation the FPGA thread searches for the framesync and adjusts the length of the next frame to be read. Thus the demo should recover from the effects of Windows latencies exceeding the desired length.

Images for display are double-buffered. This ensures that we always have one good image ready for display, while a second image is being received from the FPGA. Thus, should the display need updating for any reason other than a new image being available, the old image can be redrawn. This covers the various screen updates that Windows can throw in – for our window being moved, or for a screensaver for example. It also allows us to prevent the display of any images that are corrupted by the windows response issues. These issues simply cause some frames to be lost, but the previous image continues to be displayed so it appears as a small stutter only.

There is an exe file supplied on the HUNT ENGINEERING CD that is generated from the MSVCC project supplied. You can run it from the CD or copy it to your hard drive.

If you copy it to your hard drive, you need to copy the whole directory structure because the program requires some other files. The network and camera.dat files are examples of this, but it also looks for bitstream files in a directory that is ../bitstreams.

To run the demo you need to double click on the file viewer.exe and you will see the window below :-



As the instructions say, you need first to select a bitstream using the Setup menu. You need to choose a bitstream that matches your module type and camera type.

The /bitstream directory contains two sub-directories, one for Camera Link camera bitstreams and one for RS422 bitstreams. Inside each of these directories are directories for each FPGA module type supported.

For the Camera Link directories, there are two options for each particular modular type. For cameras with Pixel Clocks of 24MHz or below, the file that ends '_lf' (low frequency) must be used. For cameras with Pixels of 25MHz or above, the file that ends '_hf' (high frequency) must be used.

Next you need to set the module slot where your module is fitted, using the Setup-> Slot

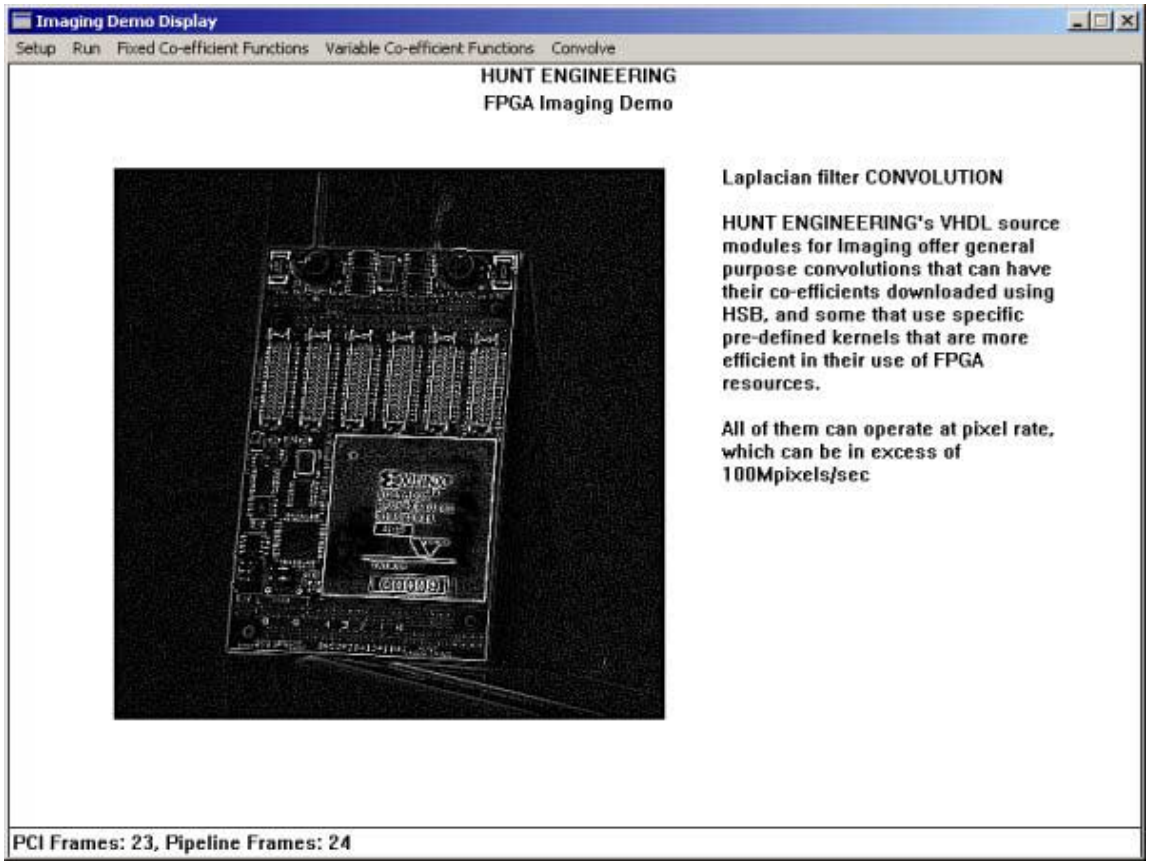
menu.

You can select the SDRAM and Self Running options now or later – as you prefer.

Next choose Start from the Run menu. The program will load the bitstream into the FPGA, and set the FIFO connections between the module and the PCI interface. Then images will be captured and displayed without any processing.

You are now free to select the various image processing functions from the other menus. The functions under the menu Variable Coefficient Functions will be greyed out unless you select the SDRAM option under Setup, and will not work unless you have a module that provides SDRAM.

If you select Self Running the functions will cycle around in a repeating demo like you would see at a tradeshow.



The file image_defs.h defines the image size used in the demo to be 384pixels by 384 lines. It will select that size of image from any camera image that is larger than that size.

If you change the definitions in this file and re-compile the host application, the host application and image acquisition will work with the new size – as long as the camera is providing an image at least as large as you select.

However for efficiency of FPGA resources, the line length is determined at synthesis time of your FPGA design. If you want to use the demo with a different size image you will need to change the definition of LINE_LENGTH in the user_ap1.vhd file and re-create the bitstream.

The example is supplied as a Microsoft Visual C++ project. This contains the following directory structure:

Directory	Description
.\	Root directory for the project; contains project management files. Also contains documentation for this example.
.\host_code	PC code for controlling the system and displaying the images. Details of the files are given below.
.\bitstreams	The standard bitstreams supplied for the various combinations of module and camera type..
.\fpga5	The FPGA project for an HERON-FPGA5 module. You can use the user_ap level and below in a project for another module type

Host Code Directory

File	Description
Viewer.exe	Main PC application.
Img_Display.cpp	Source code for demo. Note that although this is a .cpp file, it is almost straight C.
Img_Display.rc	Resource file for the demo. This file defines the menus that appear on the application's menu bar.
Resource.h	Resource header associated with the resource file.
Image_defs.h	Definition of the image size
Global.h	Global definitions for HSB addresses etc
Msg_defs.h	Definitions of values used in HSB messages
Demo_defs.h	definitions of the demo text strings

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.