

PCI DV CLS Addendum: Using the send_tiffs application with the PCI DV CLS Simulator

Engineering Design Team, Inc.
2010-04-14

CHANGES

- 2005-06-07: Created.
- 2005-06-15: Added CODE OVERVIEW section.
- 2009-02-05: Updated COMMAND OPTIONS section
- 2009-10-15: Added RGB emulation.
- 2010-04-14: Created PDF from README

Introduction

send_tiffs is an example / demonstration application that shows how the EDT library can be used to send images through the EDT Camera Link Simulator. This example uses tiff format as input but programmers can implement other image formats by linking the appropriate libraries and modifying the code as needed.

The Simulator card can be configured so its output matches the output of most cameras. For more information on that, see the CLSIM manual.

The program reads a list of images and sends each one through the simulator. The images can be any size up to 4096 pixels wide by 15000 lines.

The PCI DV CLS board can be configured for 1, 2 or 4 output taps. 3 taps (RGB) is not explicitly supported by the board, however it is emulated in *send_tiffs* by sending 4 taps with the last one as a dummy. This has the same effect at the receiving end as RGB; the caveat being that the board will DMA 32 bits instead of 24; therefore it takes 4/3 the bus bandwidth as true RGB and limits the max pixel clock speed that the board can be run to 45-50 MHz (66MHz or faster PCI busses) or 22-25 (33 MHz busses).

References

See the PCI DV CLS Users Guide and PCI DV API available for download from www.edt.com.

Installation

Your installation of the EDT software should include the source and binary versions of this program.

Operation

Before running this program, the simulator card needs to be initialized and set up. That is accomplished by using the simulator initialization program, *clsiminit*, which loads the *clsim.bit* firmware file into the interface FPGA on the simulator card, and will set up the simulator based on any given camera configuration file. For example,

```
C:\EDT\pdv> clsiminit -f sim_4096x1280.cfg
```

will set up the simulator according to the parameters specified in the *sim_4096x1280.cfg* camera configuration file. See `clsiminit --help` for more help with that program.

The simplest method of running *send_tiffs* is to run it without any arguments:

```
C:\EDT\pdv> send_tiffs
```

send_tiffs looks in the file *imagelist* in the current directory by default and reads in a list of TIF images which will be sent through the simulator card. The format of the list is given below in detail (see SPECIFYING IMAGES), but it can simply be a list of filenames, one per line.

send_tiffs will then open the unit 0 card which should be the simulator, and then the program will loop through the list of images one at a time and send each image.

If options such as unit number, *imagelist* file, number of buffers, etc. need to be specified, see the Command options section.

Command Options

Running *send_tiffs --help* will display the command options:

```
C:\EDT\pdv> send_tiffs --help
usage: ./send_tiffs [-u unit] [-N #buffers] [-i file]
-u unit          unit number of CLSIM (default 0)
-l loops        loop through image set "loops" times (or 0 for forever)
-N #buffers      number of ring buffers to use
-i file          input file with list of images and parameters
-t #images       number of images to send
                  (no larger than the number found in image list)
-A FillA         Set the left fill value to FillA
-B FillB         Set the right fill value to FillB
-v mask          set verbosity to debug output mask:
  1              just some basic debugging output
  2              dma
  4              image
  8              frame valid
  16             events
  32             image read
  0x80           everything
```

The `-u`, `-N`, `-l` and `-t` options take a single numeric argument. For example, *send_tiffs -N 10* will tell the program to use 10 DMA buffers. The `-i` option takes the name of the *imagelist* file, so for instance *send_tiffs -i list1.txt* will tell the program to look in the *list1.txt* file for the list of images.

All these options should be quite self-explanatory, except perhaps `-t`. This option is used in case you have a large number of images in the *imagelist* but want to send fewer than that. So if there are 2000 images in the *imagelist* you can use *send_tiffs -t 10* to just send the first 10 images from that *imagelist* file.

Specifying images

The *send_tiffs* program will read a file containing a list of images to send. The name of the file is *imagelist* by default, but can be specified using the `-i` option to the program as described above. The format of this file is now described.

Each line of the file contains either a comment or information about an image file. If the first character in the line is "#" the line is a comment and will be ignored. Otherwise the first set of characters up to a space (' ') character specifies the filename of the image file. The image file must be in the TIF format. That filename is the only required piece of information, but the following can also be specified:

`FillA:Value`

Sets the fill value for the left margin of the image as described in the PCI DV CLS manual. Note that specifying `-1` will tell the program to use the pixel value found in the first pixel of the first line in the image as the margin value.

`FillB:Value`

Sets the fill value for the right margin of the image as described in the CLSIM manual. Note that specifying `-1` will tell the program to use the pixel value found in the last pixel of the first line in the image as the margin value.

`hStart:Value`

The value specifies where in the camera's image to place the image from the file. For example, a line with `image.tif hStart:600` will place the first pixel of the *image.tif* image at 600 pixels into the image seen from the camera. If `hStart` is not specified, the default behavior is to center the image file within the camera's image. That behavior can be specified manually by setting `hStart` to `-1`. (`image.tif hStart:-1`). Look for `DEFAULT_HSTART` in the source code to change this default.

`vgap:Value`

The value specifies how much vertical gap to have between the current image and the next image. If not specified anywhere, the value used is `DEFAULT_VGAP` in the source code.

Note that the value is always a numeric value specified in either decimal or hexadecimal. To use hexadecimal, just put `0x` before the number. Note also that for any image file listed, values not specified will be taken from the last line on which they were specified or (if they were specified nowhere) from the program defaults as described for each parameter. Finally, the capitalization of the parameters doesn't matter, but it is necessary that the value follows immediately after with no space in between.

Source code overview

The very first thing done by `send_tiffs` is parsing command line arguments, getting the list of images, and doing some initial setup such as creating the EDT ring buffers and getting the simulated camera's size (as set by `clsiminit`).

Next, `send_tiffs` pre-loads all but the last ring buffer with the first images from the image list.

The main loop of the program only has four things to do:

- 1) Start DMA to send the image data to the simulator.
- 2) Ensure the simulator gets setup for the *next* image that will be sent.
- 3) Get another image from the list and load it into the buffer that just finished being sent to the simulator.
- 4) Wait for the image we started sending at step 1.

The settings (width,height, etc) for an image can be set on the simulator any time before the simulator begins sending that image. That is, any time before the simulator sets Frame Valid high.

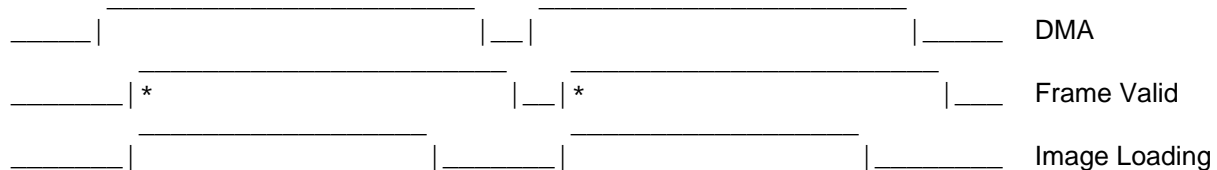
Previously, step 2 was accomplished by watching a bit on the STATUS register of the simulator, which tells us if Frame Valid is high. Only when Frame Valid is high would we set up for the next image. This was problematic because there is no guarantee that we have not missed Frame Valid going high then low. The new method is to use the EDT Event system, which makes the code much simpler & ensures we never miss images.

With the EDT Event system, we use the following :

```
edt_set_event_func(pdv_p, EDT_PDV_EVENT_FVAL,
    (EdtEventFunc) setup_clsim_event, &cbinfo, 1);
```

to register a function (`setup_clsim_event`) which will be called when the driver receives an interrupt notifying it that Frame Valid went high. So as soon as frame valid goes high it is possible to setup the simulator for the next image, and that is exactly what `setup_clsim_event` takes care of.

Although the Event system is better, it may not be clear how the timing works in the program now. Here is a timing diagram showing when different things are happening and for how long. (If the diagram doesn't look readable, try using an editor with a fixed width font, like "edit" on windows or VI on Unix).



* = Setup for next image happens at rising edge of Frame Valid

The DMA starts with the program's call to `edt_start_buffers(pdv_p, 1)`. After the simulator has gotten 16Kbytes of data it will start sending that out to the frame grabber, and at the same time it will send the FVAL interrupt to the driver which will in turn call the `setup_clsim_event` (see the asterisk * in the diagram). Immediately after the program calls `edt_start_buffers` it will load the next image into the buffer most recently sent (so the buffer being loaded is just behind the buffer being sent). Note that the image loading should take less time than the DMA transfer for maximum speed.