Applicationguide, VGADemo

This demo demonstrates how to map a framebuffer in the FPGA to the memory space of a user-mode application, how to write to the mapped framebuffer, and how to output the framebuffer contents to the VGA port of the Spartan3 PCI Express Starter kit.

Files used

The files used for this application can be found on CD1 in:

- FPGA Design: "*Source\FPGA_Design\VGADemo*"
- Windows device driver and application: "*Source\Windows_Device_Drivers_and_Applications\VGADemo*"

The FPGA design

The *VGADemo* FPGA design builds on the structure described in journal 2. The design uses a 24 kB dualport blockram at BAR0 as a framebuffer, allowing for a 128*128 image with 12 bit colors to be stored. An additional module reads the framebuffer and outputs it on the VGA port.
The framebuffer uses a 14 bit address bus, and a 12 bit data bus (2^14 x 12). In the PIPE core, BAR0 is specified as a 64 kB memory BAR (2^14 x 32), to obtain a correctly sized addressing space. When writing to the buffer, the upper 20 databits are simply discarded. Only writing to the framebuffer is supported, reading any address will just return 0.
The *gfx* module handles the VGA port, through the videoDAC mounted on the Spartan3 PCI Express Starter kit. Both the *gfx* module and the videoDAC are connected to a 25.175 MHz onboard clock, which provides support for generating a VGA signal at a resolution of 640*480 with a 60 Hz refresh rate. This is usable even though the framebuffer can only store 128*128 pixels, as the front- and back-porches[1] (which are used to constrain the valid image area of the VGA signal) are adjusted to allow only the centre 128*128 pixels to be enabled, the rest of the image will just be displayed as black.
The module uses two counters, a horizontal and a vertical, to run through every position in the image. When inside the valid 128*128 area, the difference between the counters and their respective front-porch is used to address a pixel value in the framebuffer. The pixels are stored in RGB colors, using 4 bit for each color. The retrieved pixel is split up, and each color is sent to the respective input of the videoDAC.

The driver

The driver used is just a renamed version of the *Empty* driver.

The application

This application is Windows-only, but porting it to Linux should be trivial.
The application first opens and creates a handle to the driver. It then reads in a file, specified as an argument when calling the application, to an internal buffer, *imgBuffer*. For this to work correctly, the file should be a 128*128 pixel uncompressed .raw image file with 24 bit RGB colors and no header (like *test.raw* found alongside the application). The framebuffer at BAR0 is then mapped to the applications memory space, as the ULONG pointer *map*. To write the image to the framebuffer, a for-loop runs through every pixel in *imgBuffer*, converts this to 12 bit color (by throwing away the 4 least significant bits for each color), and then stores this in the appropriate location in *map*.

[1] An introduction to VGA-timing can be found in the document "*VGA Timing.pdf*" in the *Misc* folder on CD1.

Usage

The demo is ready to use, simply connect a monitor to the VGA port of the Spartan3 PCI Express Starter kit, load the FPGA design into the FPGA (synthesize/implement if necessary), install the VGADemo driver, and run the application with the following syntax:

`vgademo` *`image_filename`*