Application Note
VHDL frame reader and VGA syncer

Authors:
frame reader - Billy Kozak,
VGA syncer - Jeff Theriault, Billy Kozak

Description

The VGA frame reader and syncer are two VHDL written parts that are designed to be used together for the purpose of putting frames on a VGA monitor using the Altera DE2 development board (specifically the VGA DAC and Avalon memory mapped interface of the DE2). The syncer can be used independently but the frame reader is dependent on our unique VGA syncer (or equivalent device) in order to function properly.

The frame reader reads data from a memory device (conforming to the Avalon memory mapped slave interface) to buffer lines of pixels as fast as it can so that these values can be read by the VGA syncer device. The reader reads 16 bit values from memory using burst reads. Each 16 bit value corresponds to 1 pixel (5 bits of red followed by 5 bits of green followed by 5 bits of blue). The reader interprets the resolution of the image to be 320 pixels wide by 240 pixels tall in row major format. Once hooked up to the VGA syncer, it will be possible to output stable frames to a monitor.

The VGA syncer outputs the digital VGA signals at very precise times to create an image. It drives address lines to read data from the frame reader and output pixel values at the exact times when they are expected. The syncer also provides two special signals to communicate with the Frame Reader. One signal transitions from high to low or low to high when the Syncer has finished reading data for a row so that the Frame Reader can keep track of which row the Syncer will need next. The second signal is a signal that is held low for a brief period every time that also occurs when the Syncer has finished reading a line. This signal is used to asynchronously reset a counter that keeps track of which column the Frame Reader needs to read from next.

The Frame reader solves the problem of RAM being too slow to be addressed and read by a VGA syncer device on demand. We do this by using a combination of burst reads (using the Avalon memory mapped interface) and by taking advantage of buffering (helpful since the VGA syncer will not always be reading pixel data every cycle).

Set Up and Use

The easiest way to use our Frame Reader and syncer combination is through the use of SOPC builder. In the SOPC builder GUI create a custom part using frameReader.vhd. The signals should be automatically defined. Under the interface tab set the input “clk” to be to clock associated with the “inter” interface. Click finish and the part should generate without error. Do the same to create the vga_sync.vhd part. Then, instantiate each part. In the GUI window for the Frame Reader there will be a place to enter the base address of your frame reader. After this

connect the master output of the Frame Reader to the desired memory device.

Next you will need to create a clock that runs at (or close to) 25.175 MHz.

After you are done configuring the rest of the system as desired, assign base addresses and generate the system. In the system.vhd file you should find several entries for the frame reader and syncer. In the top level vhd file you will need to make the following connections:

coe_blue_export_from_the_frame_reader -> coe_iBlue_export_to_the_vga_sync
coe_red_export_from_the_frame_reader -> coe_iRed_export_to_the_vga_sync
coe_green_export_from_the_frame_reader -> coe_iGreen_export_to_the_vga_sync

coe_resetCol_to_the_frame_reader -> coe_colRST_export_from_the_vga_sync
coe_rowClock_to_the_frame_reader -> coe_video_export_from_the_vga_sync

coe_rst_export_from_the_frame_reader -> coe_iRST_export_to_the_vga_sync

coe_x_to_the_frame_reader -> coe_px_export_from_the_vga_sync
coe_y_to_the_frame_reader -> coe_py_export_from_the_vga_sync

coe_BLANK_export_from_the_vga_sync -> VGA_BLANK
coe_HSYNC_export_from_the_vga_sync -> VGA_HS
coe_SYNC_export_from_the_vga_sync -> VGA_SYNC
coe_VGAB_export_from_the_vga_sync -> VGA_B
coe_VGAG_export_from_the_vga_sync -> VGA_G
coe_VGAR_export_from_the_vga_sync -> VGA_R
coe_VSYNC_export_from_the_vga_sync -> VGA_VS

VGA_CLK -> 25.175MHz clock

coe_iCLK _export_from_the_vga_sync -> 25.175MHz clock

After compiling and programming the board you should be able to hook up a VGA monitor and see a representation of whatever is in your memory. You will be able to load the memory using another memory mapped master device and put an image on the screen.

There will be two slave interfaces (one for the Frame Reader and one for the syncer). The slave interface on the Frame Reader can be used to start and stop reading from memory. Write a 0 to the register to stop reads and a 1 to restart (the device will start automatically without any input however). This will not stop the Syncer and you will see a weird output on the monitor if you leave it stopped for a long period. Note that in our project with the Frame Reader we saw no benefit from stopping and starting the memory reads. The register on the Syncer does nothing but seems to be necessary to prevent certain buggy behavior from SOPC builder.

Some Extra Details for the Frame Reader and Syncer

The syncer makes use of Burst Reads and will try to Burst an entire (320 pixel) line from memory every time it reads. If for any reason the Reads are interrupted halfway through, the Reader will hang and the device will require a full reset to function correctly (the stop/go bit does not stop the reads until the last burst read is finished).

The syncer expects a standard 480*640 VGA resolution. It is in the Frame Reader that we divide down the resolution to 240*320. This is done by returning the same pixels twice in a row when the syncer requests two adjacent horizontal pixels. Then the Frame reader will output the same horizontal line a second time so that you get pixels that are 4 times as big.

It shouldn't be difficult to examine the source code and determine how to use the full 640*480 resolution. For our work in creating animation using the NIOS we found that 640*480 was too big for performance and memory reasons.

The Frame Reader buffers one whole 320 pixel (16 bit) line in a read buffer while the Syncer reads from the write buffer. When necessary the read buffer and write buffer will swap so that the read and write processes do not interfere. This is also how we were able to deal with the issue of using two very different clocks.

Authors:
frame reader - Billy Kozak,
VGA syncer - Jeff Theriault, Billy Kozak