Lab 4: VGA Signals and Color Raster Picture

EEL 4712 – Fall 2014

Objective:

The objective of this lab is to study the generation of the control signals required by a VGA monitor by creating a component VGA_sync_gen that generates those signals. Using VGA_sync_gen and additional components, a simple color raster image will be displayed on a VGA monitor.

Required tools and parts:

QuartusII software package, ALTERA DE0 board, and VGA Monitor.

Device(s) Used: Altera Cyclone III EP3C16F484C6 FPGA, Quartus altsyncram component, brom.mif.

An altsyncram component will be used in this lab, which requires memory that is available only on the device such as the Cyclone III EP3C16F484C6 FPGA. All output bits of this lab should be directed to the appropriate pins of the ALTERA DE0 VGA connector. Also, a memory initialization file (brom.mif) will be used to initialize the memory values of the ROM.

References: ALTERA DE0 Board Manual

Introduction:

The VGA monitor connector on the ALTERA DE0 board consists of five signals, RED, GREEN, BLUE, HORIZ_SYNC, and VERT_SYNC. Look into the ALTERA DE0 Board User Manual for the header pin outs for the ALTERA DE0 board for these 5 signals. The timing relationships among this signal are shown in **Figures 1 and 2** at the end of this lab document. The generation of signals needed for the raster on the VGA monitor begins with dividing the frequency of the 50 MHz clock on the ALTERA DE0 board down to a 25 MHz pixel clock which is then used to generate the horizontal and vertical sync pulses using a ***horizontal counter*** and a ***vertical counter***.

Horizontal and vertical synchronization. A video display consists of 640 pixels in the horizontal direction and 480 lines of pixels in the vertical direction. The monitor starts each refresh cycle by updating the pixel in the top left-hand corner of the screen, which can be treated as the origin (0,0) of an X–Y plane. After the first pixel is refreshed, the monitor refreshes the remaining pixels in the row. When the monitor receives a pulse on the ***HORIZ_SYNC*** pin, it refreshes the next row of pixels. The time required for the sweep, the horizontal sweep period, is nominally 31.77 μs. This process is repeated until the monitor reaches the bottom of the screen. When the monitor reaches the bottom of the screen, a 64 μs pulse applied to the ***VERT_SYNC*** pin, causing the monitor to begin refreshing pixels at the top of the screen (i.e., at [0,0]). As shown in Figure 2, the VERT_SYNC pulse must be repeated every 16.6 ms (vertical sweep period). A complete screen of information is being traced by the electron beam every 16.6 ms for a frame rate of 60 Hz.

Blanking intervals. In order to accommodate the time required to move the electron beam back to the left side of the screen (horizontal retrace period) the electron beam must be turned off during the retrace time or the return trace would be visible. This is accomplished by two blanking intervals during the horizontal refresh cycle marked by B and C at the beginning of the trace and E at the end of the trace in Figure 1. Similar blanking intervals are required during vertical refresh (P, Q, and S in Figure 2). The color beams are turned on only when the horizontal counter is in the period D and the vertical counter is in period R.

Hint: Blanking intervals can be easily implemented by creating a video “gate” signal (Video_On) that is true only when the color beams can be active. Refer to the figure 3 at the end of this lab write-up for a graphical look at the VGA display signals.

Pre-lab requirements:

1. Pixel Clock Generate Component

Create a behavioral component to generate the pixel clock (Input: clk, Output: pixel_clock). Use the pixel clock as the system clock for all the components in this lab. **DO NOT use the 50 MHz clock in any other component.**

2. VGA Sync Component (VGA_sync_gen)

Create a ***behavioral VHDL*** file for a component called VGA_sync_gen.

- Include the IEEE.NUMERIC_STD package.
- Include the work.VGA_LIB package, which is a custom package for this project provided for you in vga_lib.vhd that contains some useful constants.
- The VGA_sync_gen entity has a clock input and 5 outputs: Hcount, Vcount, Horiz_Sync, Vert_Sync, and Video_On.
- Create two counters. Both counters should be declared as 10-bit std_logic signals.
  - **Hcount**
    - Counts up to the horizontal period (H_MAX – See vga_lib.vhd) using the 25 MHz pixel clock generated from the 50 MHz clock from the on-board oscillator.
    - To use the constants provided for you in vga_lib.vhd, a value of zero on Hcount corresponds to the beginning of section D in Figure 1.
  - **Vcount**
    - Counts up to the vertical period (V_MAX – See vga_lib.vhd). It will increment at a particular point in the horizontal counters count (Hcount = H_VERT_INC – See vga_lib.vhd).
    - Again, to use the constants provided for you in vga_lib.vhd, a value of zero on Vcount corresponds to the beginning of section R in Figure 2.
- The values of Video_On, Horiz_Sync and Vert_Sync are determined by decoding the values of Hcount and Vcount and comparing the counter values to the constants provided in vga_lib.vhd.

Verify that the two counters work individually using the simulator. An entire simulation of VGA_sync_gen is probably out of the question on your computer. First focus your simulations on the horizontal refresh cycle, since it can be simulated in its entirety easily. Then try to measure the length of VERT_SYNC (64 μs) in simulation, and if time permits try to do a full simulation of the vertical refresh cycle.

Turn in on e-Learning: all design and testbench files (.vhd files)

Printout for your TA: all design and testbest files (.vhd files) and representative simulation results of both counters and VGA_sync_gen.

3. Display of Raster Picture

In this part of lab, you will partition the screen into 2x2 pixel-sized blocks, each of which displays a color made from a combination or Red, Green, and Blue. There will be a total of 4096 blocks arranged as a 64x64 grid, with the top left block beginning on the screen at (0,0). We will need to utilize **three additional parts** to implement this color raster picture: a block ROM which contains the R,G,B values for each 2x2 block, a part which generates the current block row, and a part which generates the current block column. They are described as follows:

a) **Block row counter**: This takes the Vcount singal generated in Prelab step 1 and generates a 6-bit row count which identifies one of the 64 rows of the 64x64 grid. The block row count will be used as a part of the address input to the BROM discussed below.

b) **Block column counter**: This takes the Hcount signal generated in Prelab step 1 and generates a 6-bit column count which identifies one of the 64 columns of the 64x64 grid. The block column count will be used as a part of the address input to the BROM discussed below.

c) **Block ROM**: The block ROM (BROM) is used to contain the R,G,B values for the 64x64 grid. Thus, the size of BROM is 4096x12, having a 12-bit address and a 12-bit output. The 12-bit address consists of the row (6-bits) and column block (6-bits) counts (from parts 3(a) and 3(b) above. The 12-bit output represents the color of Red (4 bits), Green (4 bits), and Blue (4 bits) for each 2X2 pixel block.

The BROM component is made from the altsyncram component provided by Quartus II. Use the Megawizard Plugin Wizard to create a 1-Port ROM (in the memory compiler section).

- Tools -> MegaWizard Plug-In Manager
- Select “Create a new custom megafunction variation”, click “Next”.
- Click on the triangle next to “Memory Compiler”.
- Click on “ROM: 1 PORT”.
- Specify that the output should be VHDL (if it is not checked already).
- Browse to the desired folder and name your file vga_rom.vhd.
- Click “Next”.

On the next page of the wizard, spccify that:

- “q” output bus should be 12 bits.
- There should be 4096 words.
- Leave everything else as default.
- Click “Next”.

On the next page:

- “q” output should not be registered.
- Leave everything else as default.
- Click “Next”.

On the next page:

- Browse to the provided file brom.mif.

Leave everything else as default and finish.The generated file (vga_rom.vhd) can now be used as a component in your design.

A memory initialization file (brom.mif) will contain the actual memory values of the ROM and can be edited by using the Memory Editor in Quartus (or using a text editor). The brom.mif can be downloaded from the class website and can be edited to modify the “picture” being displayed on the VGA monitor. Using the original brom.mif file, you should see a color gradient changing from black on the top left corner to white on the bottom right corner.

Tasks for Prelab 3: Design and simulate each part described above with limited simulation runs, using your own testbenches (simulate each component individually).

Turn in on e-Learning: all design and testbench files (.vhd files)

Printout for your TA: all design and testbench files (.vhd files) and representative simulation results of the above parts.

4. Final test setup

PORT MAP together your pixel_clock_gen, VGA_sync_gen, vga_rom, block column counter, and block row counter to make a circuit that will display a raster picture. Use one or more simulation runs to prove that your design works. Create a top level entity which has a 3-bit vector as input and generates the VGA signals as outputs. Assign the input vector to the push button switches. If the input signal is 0, show the image at the center of the screen; if it is 1, at the top left, if 2, at the top right, if 3, bottom left, if 4, bottom right (See figure 4).

Turn in on e-Learning: all design and testbench files (.vhd files)

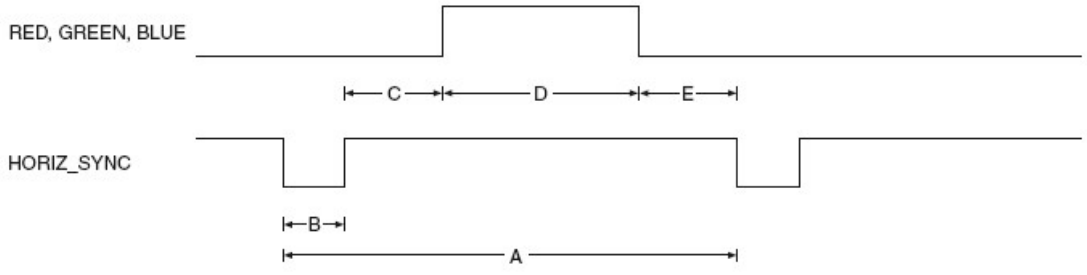
Printout for your TA: all design and testbench files (.vhd files) and representative simulation results of the final circuit.

In-lab procedure:

1. Program your board with the final test setup from Prelab Step 4 and connect a monitor to your ALTERA DE0 board's VGA connector. Confirm that a picture is displayed on your VGA monitor and show your TA.
2. Modify the brom.mif file to show a picture of your choosing. You can be creative! Some suggestions:
   - Change your brom.mif file and your Row/Column block address generators to display an image larger than 128x128 and possibly not square (eg: 200x150 pixels)
   - Animate the displayed image. Animation could be based on offsetting the address generated and using modular arithmetic or changing the video memory.
   - Extra credit will be given based on images with more complexity. A different 128x128 image is required. Anything more complex will get extra credit.
3. Your TA will ask you to modify your design in some way, so save some in-lab time for a new task.

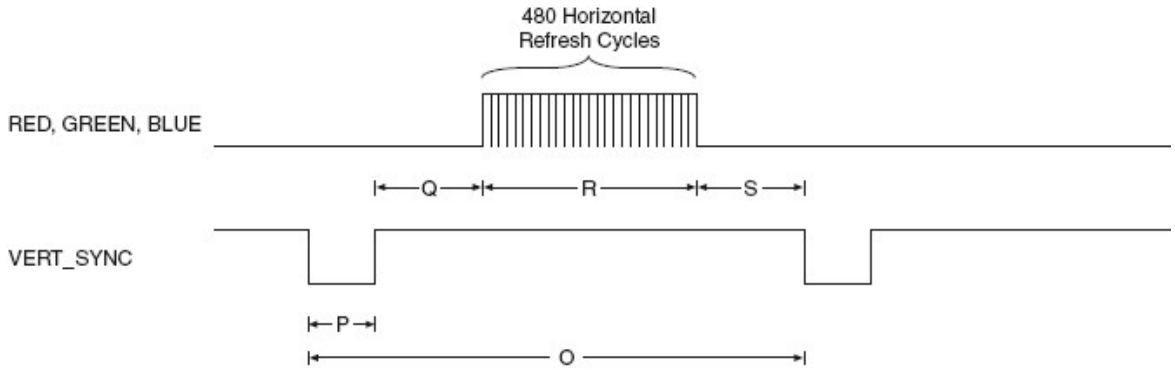
In-lab Option:

In order to test your VGA_sync_gen component you may want to design a simple component, Colorbars, which displays three vertically-aligned Red, Green, and Blue bars across the screen. This component takes Hcount as an input, and gives Red, Green, and Blue signals as outputs. It can be easily made using an IF statement. This component is not required, but will allow students to get partial credit if they cannot get their full design working.



Parameters	A	B	C	D	E
Time	31.77 µs	3.77 µs	1.89 µs	25.17 µs	0.94 µs

Figure 1. Horizontal Refresh Cycle.



Parameters	O	P	Q	R	S
Time	16.6 ms	64 µs	1.02 ms	15.25 ms	0.35 ms

Figure 2. Vertical Refresh Cycle.

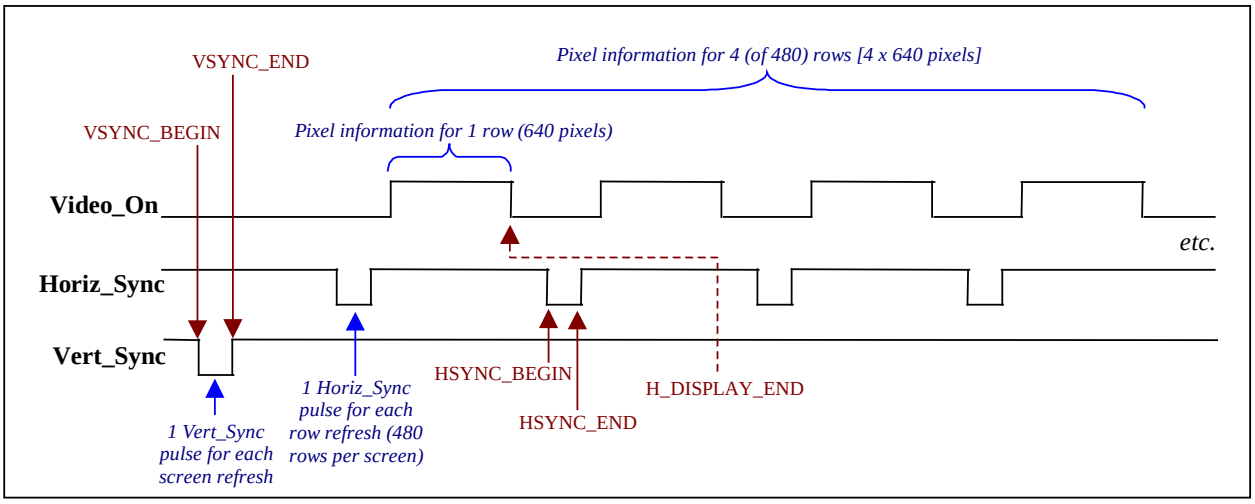


Figure 3. Timing diagram for four rows of a VGA Display.

Figure 4. Display positions for different button combinations. Exactly one of these can be active at a time.