ECE 4514 Fall 2019: Homework 6

Assignment posted on March 21
Assignment due on March 28 11:59 PM

Homework weight: 100 points

The DE1-SoC kit has a video-DAC which can generate 24-bit color images on a monitor. In this homework, you will design a simple graphics application that generates signals for a computer monitor driven through the video-DAC. The objective of this homework is threefold.

- Build insight into the graphics generation process for raster-scan displays.
- Design a Verilog implementation of a color generator and a test-pattern generator.
- Use a module-generator (from the IP Catalog) to create a specialized clock generator, and instantiate the clock generator in your design.

The homework will explain the specifications of the design, and describe a possible structure. However, the homework will reveal little on the actual design steps (writing Verilog, creating a project, mapping to FPGA, downloading the bitstream). It is up to you, as a designer, to translate the following specifications into a working project.

Design Specification

In this homework, you need to generate video images on a resolution of 1024 pixels by 768 pixels, also known as XGA resolution. The screen is redrawn 70 times per second, which results in a pixel rate of 75 million pixels per second. This means that the FPGA implementation needs to generate pixel data for the Video DAC at a rate of 75 million pixels per second.

You have to create two versions of the video generator.

1. A first version, called colors, generates a screen of uniform color. The color value is selected using the three switches SW[0], SW[1] and SW[2].

Color	SW[0]	SW[1]	SW[2]	Red	Green	Blue
Black	0	0	0	0	0	0
Red	1	0	0	255	0	0
Green	0	1	0	0	255	0
Yellow	1	1	0	255	255	0
Blue	0	0	1	0	0	255
Purple	1	0	1	255	0	255
Cyan	0	1	1	0	255	255
White	1	1	1	255	255	255

2. A second version, called testpattern, generates a test pattern of vertical colored bars on the screen. Each bar is 128 pixels wide, 1/8 of the width of the screen. The order of the colors is the same as in the table above: black, red, green, yellow, blue, purple, cyan, white.

The entire design must be made in Verilog, using a meaningful structural decomposition of the design in various sub-modules. Since the pixel rate of this design is faster than the standard 50MHz clock on the DE1-SoC, you will need to synthesize a fast clock on the FPGA, using a PLL module. This module can be created as a IP Core hard-macro, and inserted in your design.

To test this homework, you will need to make use of the VGA output on your DE1-SoC board. You will also need to make use of a screen monitor to test your design. I am checking if the CEL has monitors available in case you do not have access to a monitor; I will post an update on the Piazza if that is the case.

Implementation Guidelines: XGA Video Signal

When counting pixels on a canvas, it's common to call the upper-left pixel (0,0) and the bottom-right pixel (1023,767). The X axis then runs from left to right, and the Y axis runs from top to bottom. The drawing of a picture on the canvas is done row-by-row, from left-to-right and from top-to-bottom. Thus, in our coordinate system, pixels are generated as follows.

```
while (1) {
  // draw frame
  for (y=0; y<768; y++) {
    for (x=0; x<1024; x++) {
      draw pixel(x, y);
    }
    generate horizontal synchronization signal;
  }
  generate vertical synchronization signal;
}
```

At the end of each line, a horizontal synchronization signal is generated. This signal informs the screen hardware that the current line is finished, and that new pixel data should be displayed at the start of the next line. Similarly, at the end of each frame (after the last horizontal line), a vertical synchronization signal is generated. This informs the screen hardware that the next line to be drawn will be located on top of the screen.

In previous generations of screen technology (Cathode Ray Tube), the synchronization signals were used to control an electron beam scanning a phosphor screen. This required synchronization signals with precise timing (period and duration). Modern display technologies (LCD) process the synchronization signals entirely digital. However, for backward compatibility, video synchronization signals still have the same precise timing constraints.

Before you will be able to implement the assignment, you have to study the basics of a XGA monitor signal, and the implementation of the monitor signal on a VGA connector. The User Manual for the DE1-SoC board describes the VGA interface in section 3.6.6 on pages 32-35. Of particular interest is the VGA timing specification on Page 34. The Intel FPGA generates the following data for the Video DAC:

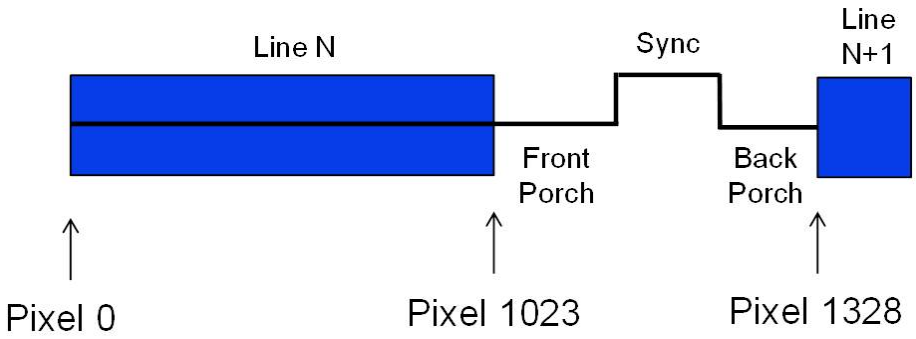
- VGA_R, VGA_G, VGA_B: 3 times 8-bit color pixel data.
- VGA_CLK: Pixel clock, 75MHz for XGA at 70Hz frame rate.
- VGA_SYNC_N}: Composite synchronization signal (negative assert).

- VGA_BLANK_N}: Video blanking signal (negative assert).

Using these signals, the video DAC generates analog R, G, B data, which is directly provided to the display monitor. The monitor combines the analog pixel data with digital synchronization signals VGA_HS and VGA_VS coming from the FPGA.

Some monitors do not have separate inputs for horizontal or vertical synchronization signals, but accept only R, G, B analog data. In that case, the green signal (G) is used to carry synchronization data as well. The video DAC is capable of mixing a {\em composite} synchronization signal into the G signal stream. The composite synchronization signal, in turn, is the XOR of the horizontal and vertical synchronization signal.

The timing specification of the XGA Video is described on page 52 of the DE2-115 user manual. This timing specification describes the relationship of pixel data and the horizontal and vertical synchronization signal. There are 4 periods to each video line: a data area, a 'front porch', a sync pulse, and a 'back porch'. For video frames, there are 4 similar areas: a data area, a front porch, a sync pulse, and a back porch. The timing of each of these sections is important.



Implementation Guidelines: 75MHz Clock Generator

In order to generate a pixel rate of 75MHz, a clock generator of at least 75MHz is needed, which is above the 50MHz signal available from the DE2-115 board.

You need to make use of a specialized circuit on the Intel FPGA to generate the faster clock. This circuit, a phase-locked loop, can not be inferred by writing standard Verilog. Instead, it must be specifically inserted from a library during the synthesis process. Intel calls these library modules IP Cores. Technically, these modules are inserted as Verilog black-box modules with an Intel-specific type name, and with specific module parameters. The Intel synthesis tools recognize the type name and parameters, and steer the mapping process.

The module that you need to create a 75MHz clock out of a 50MHz clock is called ALTPLL, and it is created as follows in Quartus.

1. In Quartus, select Tools – IP Catalog.
2. In the IP Catalog, find and select the ALTPLL module (under Clocks, PLLs and Resets). *Note – Intel requires Altera to rename everything as 'Intel'. The newer versions of Quartus (18.x) call this module Intel PLL.*

3. In the dialog, select an output filename for the module that will be created. For example, call it clockgen.v.
4. The following dialog is specific to the PLL module, and provides a wealth of options. You need to change only two things. First, select the input clock to be 50MHz. Second, select the output clock to be 75MHz. All the other options can be left at their default value. Press finish.
5. You can then add the PLL to your design project. Note that the top-level of the PLL is quite simple. Open the file clockgen.v in a text editor to inspect it. The input clock signal is inclk0, and should be connected to the 50MHz clock available on the DE1-SoC kit. The output clock signal is c0 and should be used as your system clock. The reset port can be tied to ground, and you can connect the locked output to a LED to verify that the PLL generates a reliable 75MHz clock signal.

```
module clockgen (
      input  wire  refclk,   //  refclk.clk
      input  wire  rst,      //   reset.reset
      output wire  outclk_0, // outclk0.clk
      output wire  locked    //  locked.export
);
```

What to turn in

Download the repository for Homework 6. You will find two projects: colors and testpattern. You need to create a design for each, making use of multiple Verilog modules and IP Cores as you see fit.

- Start with the colors design:
    - Generating the 75 Mhz clock generator. The altpll module will generate the internal system clock for the FPGA. If you add additional modules to the design, they will have to be clocked at the 75MHz rate.
    - Next, design a VESA sync generator, which generates the proper horizontal and vertical blanking signals.
    - Add a color selection for RGB to these two modules (clock generator and VESA sync generator). This is easy, since the entire screen uses a single color. Hence, you can use the position of the switches SW[0], SW[1], and SW[2] to directly
- Next, move to the testpattern design:
    - You can reuse most of the work of the colors design. The main difference is in the design of a pixel generator for R, G and B colors. In order to generate color bars, you will have to count x and y pixels, and select the proper R, G, B color based on the (X,Y) pixel position of the screen.

When both of these designs are finished, push your repository back to github. Make sure to include all Verilog files.

Good luck!