Microprocessor-Based Systems (E155)

Harris and Wang **Fall 2011**

Lab 7: USB, SPI and VGA

Requirements

Build a system with the PIC and the FPGA that connects to a USB mouse, tracks its movements, and displays a cursor on a VGA monitor over a background pattern. The two devices should communicate via SPI.

Overview

The PIC32 has an onboard peripheral for USB communications that is connected to the USB ports on the top of your board. The FPGA has dedicated connections to the 3-channel digital-to-analog converter (DAC) and the 15-pin D-sub connector on the board that are intended for output to a VGA-compatible monitor. Using both chips, you can construct a system that displays a mouse cursor on a screen.

Writing all the code for USB connection and VGA output in a single lab would be a Herculean task (even by Mudd standards). Instead, this lab will focus on using and adapting existing code to suit your needs. On the course web site you will find a lab7base.zip file with a working USB stack and receiver program for a class-compliant USB mouse, and a working VGA output module. Your task is to learn how these programs work and add the necessary functionality to meet your objectives.

USB Setup

Microchip provides numerous example programs for their products on their website that developers can use, free of charge, as starting points for their own work. The USB mouse code provided is a modified version of the USB mouse demo from the Microchip Application Libraries, adapted for our board and with unnecessary features removed.

Fire up MPLAB and open the project file in the usbmouse directory. The project has a multitude of files included in order to implement the full USB stack. Together, they contain over 9,000 lines of code. Thankfully, the only file you will be modifying is the top-level ‘Mouse_demo.c’. Read through this file and make sure you understand how the program works. You may need to look at specific parts of the USB stack code to clarify how data is transferred within the program. The Edit - Find in Files function may be useful to you at this step. It searches through all files in the project for a given phrase, and displays results in the Output window. Double-click a result to automatically open the appropriate file and jump to the correct line, as you would to track down a syntax error reported to the console.

Build the project and program it on to the PIC. Set your USB Mode switch on the board to 'HOST'. This connects the VBUS signal directly to the +5V input to the board to power attached devices. The VBUS signal is shared between the HOST and DEVICE connectors on the board, so be sure only one or the other is used at any given time. Additionally, be sure your bench supply is set precisely to 5V, as the supply to the USB port is not regulated.

The X and Y data reported from the mouse are the change in position on each axis since the previously reported sample, expressed in two's-complement format. Add some code to keep track of overall cursor position. Keep the cursor position within the bounds of a standard VGA screen (640x480 pixels); if the mouse reaches the edge, it should not move further.

VGA Setup

Open Quartus and create a new project in the 'vga' directory. Add the provided Verilog source to the project. This source contains an almost-fully-functional VGA output program. Read through the code so that you understand how it works.

The VGA program makes use of one of the FPGA's onboard PLLs to turn the 40 MHz system clock into the 25.175 MHz clock needed for VGA output. Launch **Tools – MegaWizard Plugin Manager**. Choose **'Create new custom megafunction variation'** and click Next. At the next dialog, expand the I/O folder in the megafunction list pane and select **ALTPLL**. Pick a name for the output file (something like 'pll' works well) and click Next.

The ALTPLL MegaWizard tool will launch. Select Speed Grade 8 in the General section, and enter the input clock frequency of 40.0 MHz. Leave the other options at their default. Click the '3: Output Clocks' tab at the top of the screen. Under Clock Tap Settings, enter the requested clock frequency of 25.175 MHz. The wizard will determine the appropriate PLL settings to generate a clock as close as possible to the target frequency. Next, click the '5: Summary' tab, and check the box to generate an Instantiation Template File. Make note of the file name (eg. pll_inst.v). This file will contain the exact Verilog statement needed to instantiate the PLL in your program. Click Finish to generate the PLL configuration.

Copy the code from the Instantiation Template File into the top-level module of your program. Now all that remains are pin assignments. Refer to the board schematic for the pin numbers. Connect `red`, `green` and `blue` to the inputs to the DAC, `syncOv` to SYNCb, and `vgaclk` to CLOCK. Also connect `HSync` and `VSync`, which go directly to the VGA connector. Compile your design and program it on to the FPGA. Connect one of the old CRT monitors to the VGA port to test the output (please don't use the nice LCDs!). You should see confirmation on the monitor that the program is working.

Try changing the code that determines what appears on the display. Experiment with it until you are comfortable drawing arbitrary rectangles. Generate an attractive pattern that fully fills the screen with multiple colors.

You will soon change the code here to overlay a mouse cursor based on the position information from the PIC.

SPI and Mouse Cursor

The PIC contains several built-in Serial Peripher Interface (SPI) transceivers. SPI is one of the simplest methods of connecting digital devices using a minimum of wires and hardware. Configure an SPI module on the PIC and use it to transmit the cursor position to the FPGA. Select a transmission speed fast enough that the cursor output will not visibly flicker as data is received. Write an SPI receiver module for the FPGA that provides the cursor information to the rest of the system.

Display Requirement

Modify the VGA code to draw a cursor overlaying your background pattern. You may use any nontrivial cursor shape you want, such as an arrow or a letter, but not a simple rectangle. Figure 1 shows an example of an A-shaped cursor overlaying a Bayer background pattern.

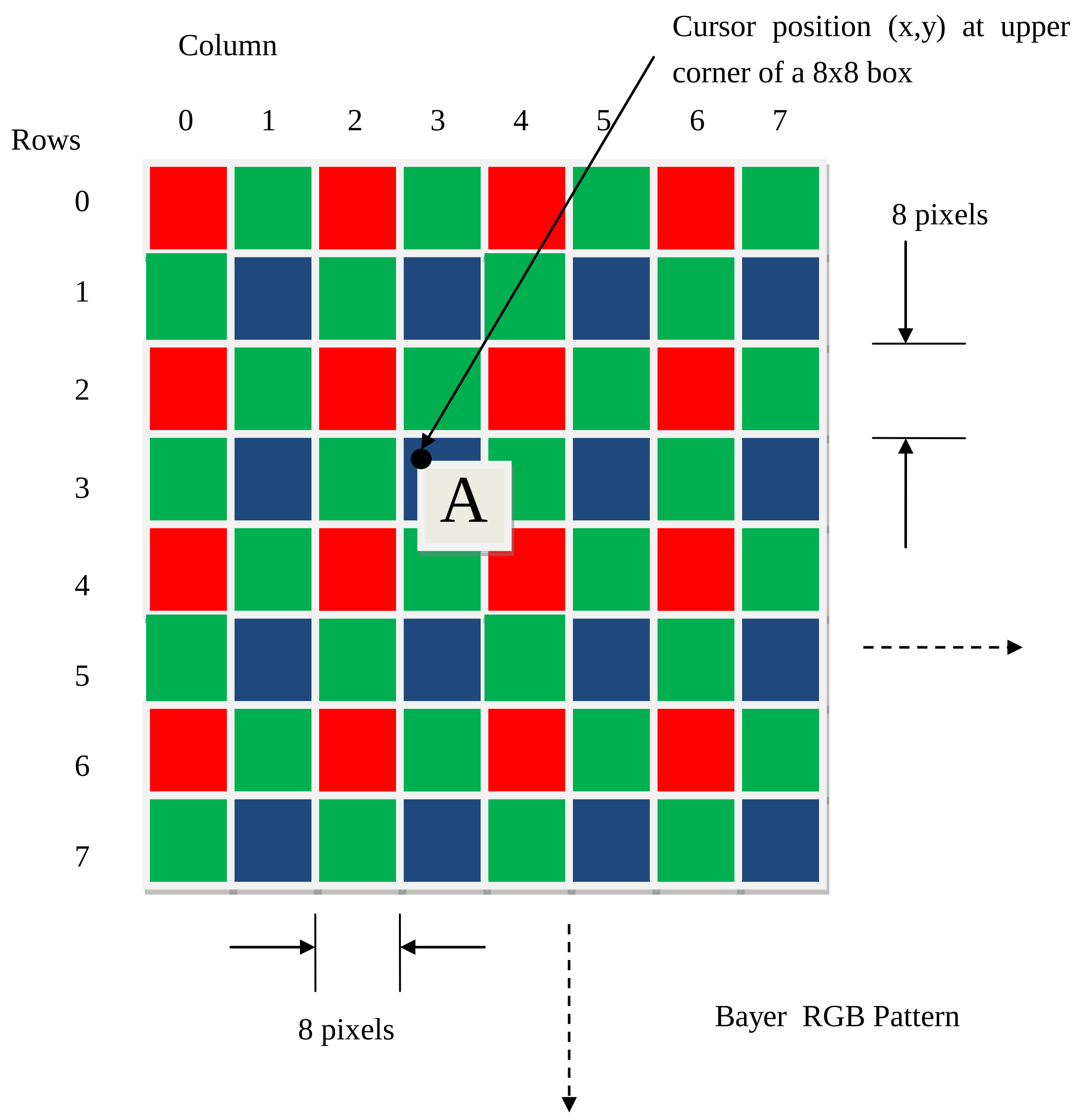

Figure 1 An example of display with a Bayer pattern as the background frame overlayed with a character at the cursor position

Hints

- Consider configuring MPLAB to automatically program the PIC after a successful build. To do this, choose Debugger • Settings • Program and check “Program after successful build.” After Program Run the program.

- Keep the SPI implementation as simple as possible. Options like framed mode and slave select are useful when connecting multiple devices to a single master, but are not necessary when using a single slave. Ideally, you should need only two signals to transmit the data. Beware of setup and hold times so that you don’t sample the data while it is changing.

- The digital inputs on the oscilloscope are useful when debugging SPI. If you are having issues, try setting the scope to trigger on the edge of the SPI clock. This can help when trying to find a short, fast signal without tedious scrolling and zooming.

- Sticking the oscilloscope probe directly into the holes in the VGA connector will be much easier than finding specific pins on the FPGA or DAC if you need to debug the monitor timing signals.

Credits

This lab was developed in 2010 by Leo Altmann ’11 and modified by David Harris and Karl Wang.

What to Turn In

When you are done, have your lab checked off by the instructor. You should thoroughly understand how it works and what would happen if any changes were made. Turn in your lab writeup including the following information:

- Your modified sections of Mouse_demo.c
- Your SPI receiver and modified sections of mouse-vga.v
- Your display design.
- How many hours did you spend on the lab? This will not count toward your grade.