COE4DS4 Lab #1
Digital System Implementation with Real-Time Constraints

Objective

To revise SystemVerilog and the Quartus design environment and learn about implementation of very simple image filters in real-time.

Preparation

- Revise the digital system design material, the Quartus design environment, and SystemVerilog
- Read this document and get familiarized with the source code and the in-lab experiments

Experiment 1

Part (a) The aim of this experiment is to get you familiarized with the Liquid Crystal Display (LCD).

LCDs are ubiquitous in today's electronic gadgets, especially in battery-powered and handheld devices, due to their small form factor and energy-efficiency. The basic principle is to regulate using an active filter the amount of white backlight that passes through it. Pixels, which consist of molecules (mostly organic compounds) layered in between electrodes, are fit into a thin flat display and are controlled through the magnitude of the Red (R), Green (G) and Blue (B) components. In this lab, the design implemented in the field-programmable gate-array (FPGA) drives the RGB values, as well as the sync signals for lines and frames. This sync signals follow the same principles as the Video Graphics Array (VGA) signals.

The design from the Cyclone FPGA will communicate with the LCD and Touch Module (LTM), attached as a daughter card to the 40-pin header on the DE2 board. There are two separate busses through which we communicate with the LCD. One of them is a *configuration bus* and the other one is the *data bus*. The configuration bus communicates with the LCD's integrated circuit (IC) device controller (available on the LTM) through a two-wire serial interface called Inter-IC bus (I2C). The two signals are clock and data, which are bi-directional. The maximum data rates achievable by I2C are around 400kbps, which are suitable for passing configuration data between devices on printed circuit boards (PCB). It is commonly used for configuring audio/video controllers, displays, RF tuners, ... The configuration data passed to the LCD device varies from the horizontal and vertical start positions to RGB gains or RGB offsets or RGB gamma correction. All the configuration registers are documented in the LTM data sheet and, in the current implementation, the communication on this I2C bus happens on power-up and is abstracted from the rest of the design. The data bus contains the RGB values, as well as the synchronization signals. The conceptual figure is given below.

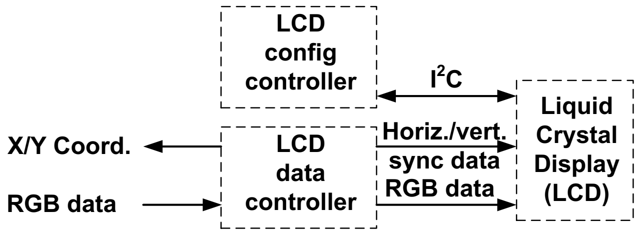

Figure 1: Interface to the LCD.

The source code given in the folder ***experiment1a*** will configure the display in the 800x480 resolution and it will display 60 frames per second. There will be 25 vertical color bars (each of them 32 pixels wide). You have to perform the following tasks in the lab for this experiment:

- Get re-familiarized with project creation, design compilation and device programming;
- Implement a revised design that displays 7.5 horizontal color bars, each of them 64 pixels high.

Part (b) The aim of this experiment is to get you familiarized with the touch panel.

The touch panel is placed in the same LTM as the LCD device (on the top of it). The communication with it is done through the same 40-pin cable attached to the DE2 board that carries the LCD signals. Touch panel's driver IC contains an analog-to-digital converter (ADC) that digitizes the position of the touch in the display area. The communication between the FPGA and the touch panel is done through the serial peripheral interface (SPI) bus. This bus contains 4 wires: clock, two data lines and chip-enable. The chip-enable is required if multiple SPI-compatible slaves are connected through tri-state devices to the same master. Due to its full-duplex capability, the SPI is capable of achieving data rates in the range of a few Mbps. This makes it suitable to transfer data from/to ADCs/digital-to-analog converters (DACs), as well as between micro-controllers and hardware accelerators, such as digital signal processors.

The configuration of the touch panel is abstracted from the user's design. What the user receives through the SPI interface are the coordinates of the touch point. These coordinates will be two 12-bit data values (X and Y coordinates), which are synchronized with 2 control signals: one that indicates if a change of position has occurred (the new X and Y coordinates will also be passed synchronously with this signal); the other indicates whether the panel is touched (note, while keeping the "finger" in the same position on the touch panel, the touch point will change continuously due to the noise in the transducer and/or in the analog signal and the resolution of the ADC). The conceptual figure is given below.

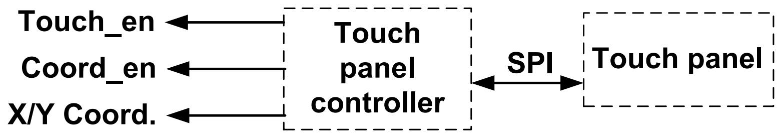

Figure 2: Interface to the touch panel.

The reference design provided to you displays the X and Y coordinates on the 7-segment displays. You have to perform the following tasks in the lab for this experiment:

- Get familiarized with the reference design;
- Modify the information displayed on the 7-segment display as follows. The touch panel can be divided into 4 equally-sized regions, called top-left (0), top-right (1), bottom-left (2) and bottom-right (3); display on the rightmost 7-segment display the code of the region which was touched. If the panel is not touched, then display 8 on the 7-segment display.

Experiment 2

The aim of this experiment is to get you familiarized with the digital camera and how it is interfaced to the SDRAM, as well as how the SDRAM is interfaced to the LCD display. The entire setup is capable of streaming image data in real-time from the digital camera to the LCD display.

Digital cameras use image sensors for optical to electrical signal conversion. The image data is passed from the digital camera to the DE2 board through a 40-pin header. As it was the case with the LCD display, we have two separate busses: one for configuration (through I2C) and one for raw image data. Through I2C we can program different parameters of the digital camera, such as image size, frame rate, color (gain, offset, gamma correction, ...). The only parameter configurable through the switches in our design is the exposure time, which indirectly affects also the frame rate.

The raw image data is buffered into the external SDRAM, which has a capacity of 8 Mbytes. The SDRAM controller takes care of generating all the address and control signals through which we access the SDRAM. What the user provides is a read enable, requesting a 32 bit word from the SDRAM. From these 32 bits we have 30 bits of image data (the digital camera generates 10 bits per color). Before passing the data to the LCD controller, for each of the color components we truncate the 10 bits to 8 bits, by discarding the 2 least significant bits. Note, the image that is displayed is in 640x480 format. The conceptual figure is given at the top of the next page.

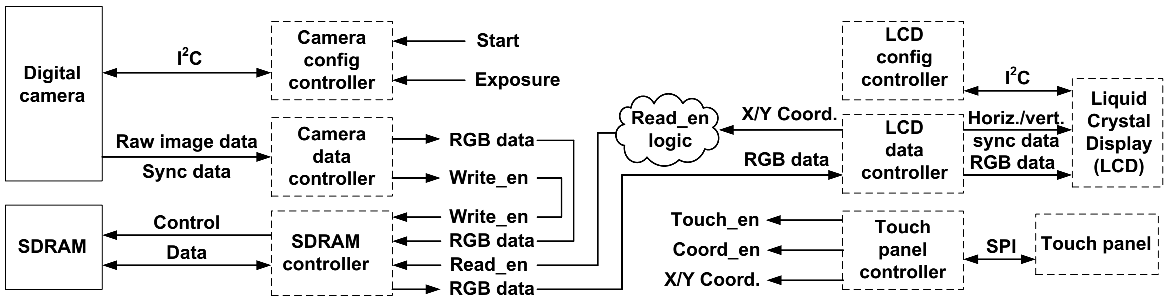

Figure 3: Interconnection of the SDRAM to the digital camera and the LCD.

You have to perform the following tasks in the lab for this experiment:

- Get familiarized with the reference design – note: SWITCH_I[17] is used as an active low reset; SWITCH_I[15:8] set the exposure time of the camera, SWITCH_I[0] chooses to display either the frame count (on) or the touch point location (off) on the seven segment displays, and three push-buttons [3:1] are used for: start video capture (3), halt video capture (2) and load exposure time (1);
- The image is displayed in color (despite the artifacts caused by the color gain and offset, as programmed in the digital camera device). You are asked to convert this color image to a gray-scale image by generating the luminance (Y) out of the RGB components that come from the SDRAM. The arithmetic equation is: Y=(1052*R + 2064*G + 401*B)/4096.

Experiment 3

The aim of this experiment is to implement a real-time low-pass filter on the image that is acquired by the digital camera and displayed on the LCD.

In the previous experiment you have implemented RGB to Y conversion. In this experiment we want to display this image either in its original form or in its smoothened form. To achieve smoothening, we apply a low pass filter on the image by averaging consecutive pixels. The conceptual figure is given below.

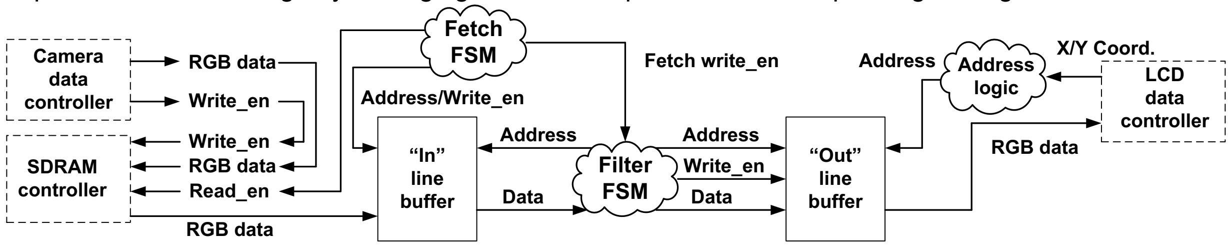

Figure 4: Real-time filtering.

The role of the "Fetch FSM" is to keep the "In" line buffer full with the next line of the image to be displayed from the SDRAM. At the same time, the "Out" line buffer passes the line currently being displayed to the LCD data controller. While the line in the "Out" buffer is being displayed, the "Filter FSM" streams the next line from the "In" buffer, through the filter and into the "Out" buffer to be displayed after the current line.

You have to perform the following tasks in the lab:

- Get familiarized with the reference design – as it can be seen from the given source code, different images can be displayed on the screen: the original (non-filtered) grayscale, the negative grayscale, … smoothened grayscale (through averaging). Signals Filter_config[2:0] are used for this purpose and they are connected to SWITCH_I[3:1];
- You are already given a low-pass filter that averages two consecutive pixels. Implement a more "accurate" filter where three consecutive pixels are interpolated to achieve smoothening. The filter structure is Y[i] = (Y[i-1] + 2*Y[i] + Y[i+1])/4. Note, Y[-1] = Y[0] and Y[640] = Y[639]. The signals Y[i-1], Y[i] and Y[i+1] are already in the Verilog source code as Y_m1, Y_0 and Y_p1 respectively. This filtered image should be displayed when Filter_config[2:0] is equal to decimal 5.

Write-up Template

COE4DS4 lab #1 report - Group #number
Student names and McMaster email addresses

There are 2 take-home exercises that you have to complete within one week. Label the top-level modules as exercise1 and exercise2. If, for any particular reason, you will add/remove/change the signals in the port list from the port names used in the design files from the in-lab experiments, make sure that these changes are properly documented in the source code.

Exercise 1 (3 marks) –Consider the touch panel and the LCD divided into four regions of equal size (called top-left, top-right, bottom-left and bottom-right).

The color of a region will change each time the touch panel has been pressed and the touch point is in the respective region continuously for a full second. The color in the top-left and bottom-right regions will change by incrementing the RGB values from 000 to 001 to … to 111 and the top-right and bottom-left regions will change their color by decrementing the RGB values from 111 to 110 to … to 000 (note, wrap around is used when the RGB reaches the max or min value). On power-up, the top-left and bottom-right regions will start from RGB=000, while the top-right and bottom-left regions will start from RGB=111.

It is worth exemplifying that if a region is pressed for 3.001 seconds, its color will change three times, i.e., each time when a full second has elapsed; however, if it is pressed for 2.999 seconds, its color will change twice, or if it is pressed for 1.999 seconds, its color will change only once; note also, if it is pressed for 0.999 seconds then released and then pressed again for 0.999 seconds, then its color will never change. Furthermore, please take note that if the touch point is moved across different regions without being released, then the same rule as above applies, i.e., the color of a region will change only if the touch point is in the respective region continuously for one full second. For example, if you press the touch panel in the top-left region and move the touching point toward the top-right region, it may happen that the touch panel is pressed continuously for 1.98 seconds, however each of the two regions is pressed continuously for only 0.99 seconds – in this case none of the two regions should update their color.

Display on the leftmost 7-segment display the code of the region which was touched (use the same 2-bit codes like in *experiment1b*). Note, however, if the panel is not touched, then the leftmost 7-segment display should not be lightened (as if the board is not powered). The four rightmost 7-segment displays show in binary-coded decimal (BCD) the number of milliseconds (ms) that have elapsed since the panel has been pressed continuously (this milliseconds counter is updated regardless whether the touch point stays in one region or it moves between multiple regions). It is fair to assume that the touch panel will not be pressed longer than 9,999 ms. When the panel is not touched, only the rightmost 7-segment display (from the four rightmost 7-segment displays) is used and it shows the decimal code of the color that appears in most regions. For the black color we have RGB=000, hence its color code is 0, for the blue color we have RGB=001, hence its color code is 1, …, for the white color we have RGB=111, hence its color code is 7. In the event of a tie-break, show the color code that has the highest decimal value. Note also, the 7-seven segment displays whose behavior was not specified above should not be lightened.

Submit your sources and in your report write (up to) a half-a-page paragraph describing your reasoning.

Exercise 2 (3 marks) – In *experiment3* you have switched between the grayscale image and its two smoothened versions. At-home, you are asked to extend this source code to support a couple of more filter configurations.

When Filter_config[2:0] equals 6 implement a simple yet effective real-time horizontal edge detector on the grayscale image. For each pixel "i", you need to compute the following formula using the grayscale intensities of the four neighboring pixels: $-2*Y[i-2] - 2*Y[i-1] + 2*Y[i+1] + 2*Y[i+2]$. The corner cases are $Y[-2]=Y[-1]=Y[0]$ and $Y[641]=Y[640]=Y[639]$. Subsequently you should perform thresholding as follows. If switch 4 is low then, if the result of the above filter formula is less than -64 or greater than 63, replace the grayscale intensity of pixel "i" with the maximum unsigned value on 8 bits (255); otherwise replace the grayscale intensity with the minimum value (0). If switch 4 is high then do the same as above, however the threshold values should change from -64 and 63 respectively to -128 and 127 respectively.

When Filter_config[2:0] equals 7 implement a real-time horizontal low-pass median filter on the grayscale image. The grayscale intensity for pixel “i” is the median value of the grayscale intensities of the four neighboring pixels Y[i-2], Y[i-1], Y[i+1] and Y[i+2] (please note, the corner cases are the same as when Filter_config[2:0] equals 6). It is important to note that since the number of samples that are used in this median filter is even, the median is the average of the two middle values. Note also, the fraction digits will be discarded when computing the average value. For example, if the four samples that are inputs to the medial filter are 2, 1, 5 and 7 respectively, the two middle values are 2 and 5, whose average is (2+5)/2=3.5, which makes the median equal to 3.

It is critical to note that an important, yet not obvious, challenge for this exercise lies in synchronizing the input and output line buffers, following the same line of thought as for the 3-tap low-pass filter done in-lab, and to be discussed in more detail in-class.

Submit your sources and in your report write (up to) a half-a-page paragraph describing your reasoning.

VERY IMPORTANT NOTE:

This lab has a weight of 6% of your final grade. The report has no value without the source files, where requested. Your report must be in “pdf” format and together with the requested source files it should be included in a directory called “coe4ds4_group_xx_takehome1” (where xx is your group number). Archive this directory (in “zip” format) and upload it through Avenue to Learn before noon on the day you are scheduled for lab 2. Late submissions will be penalized.