PixelPerfect

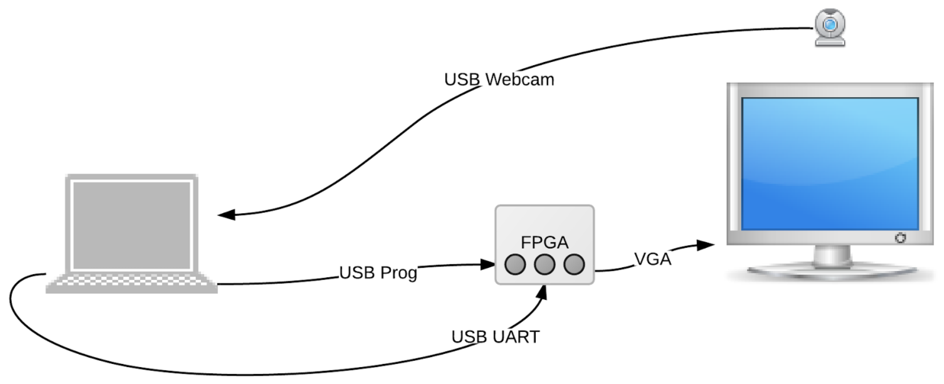
User Guide

Contents

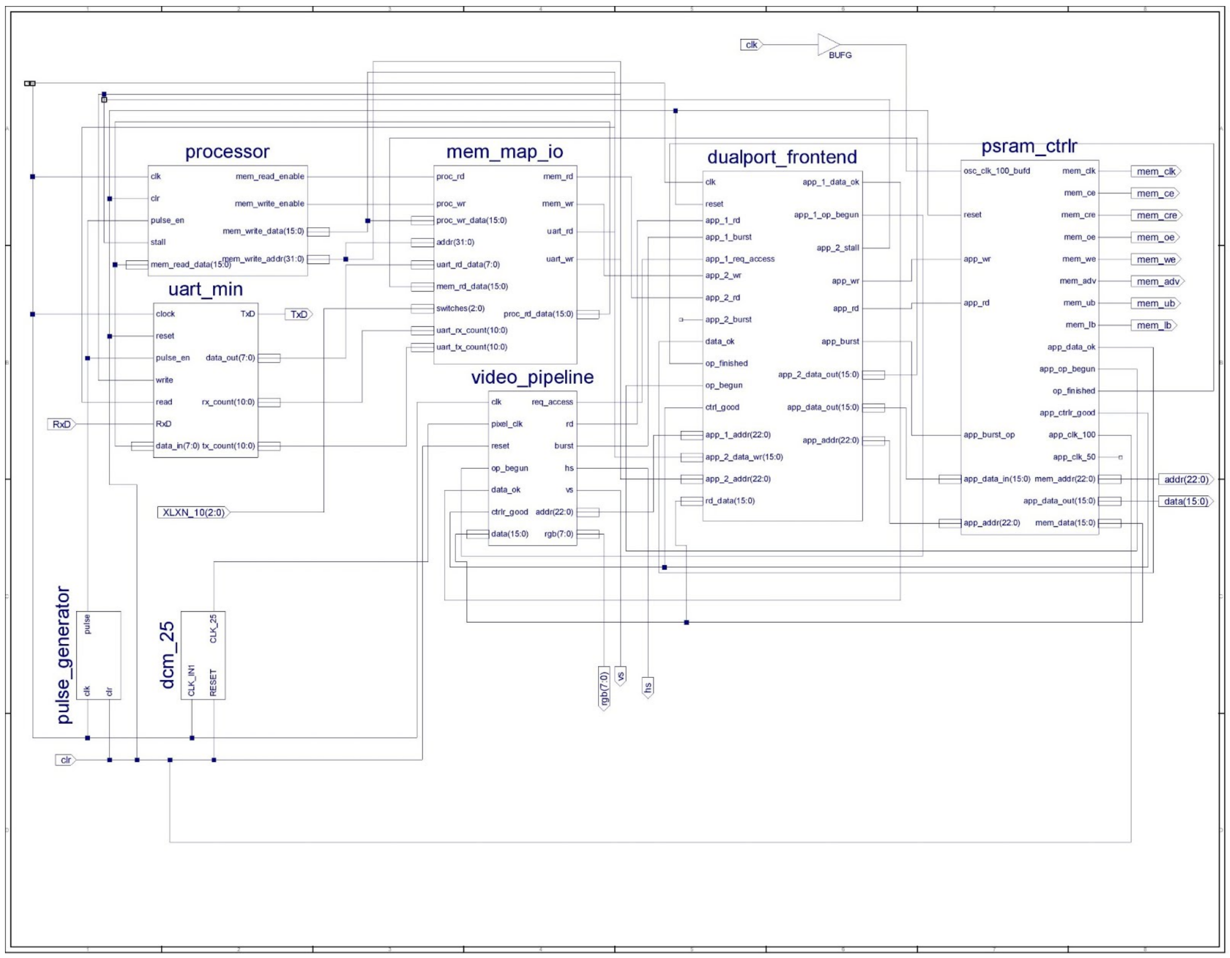
CS3710
Fall 2012
University of Utah
Kevin Avery, Scott Daw, Trevor Hill, Mac Wibbels

System Schematics

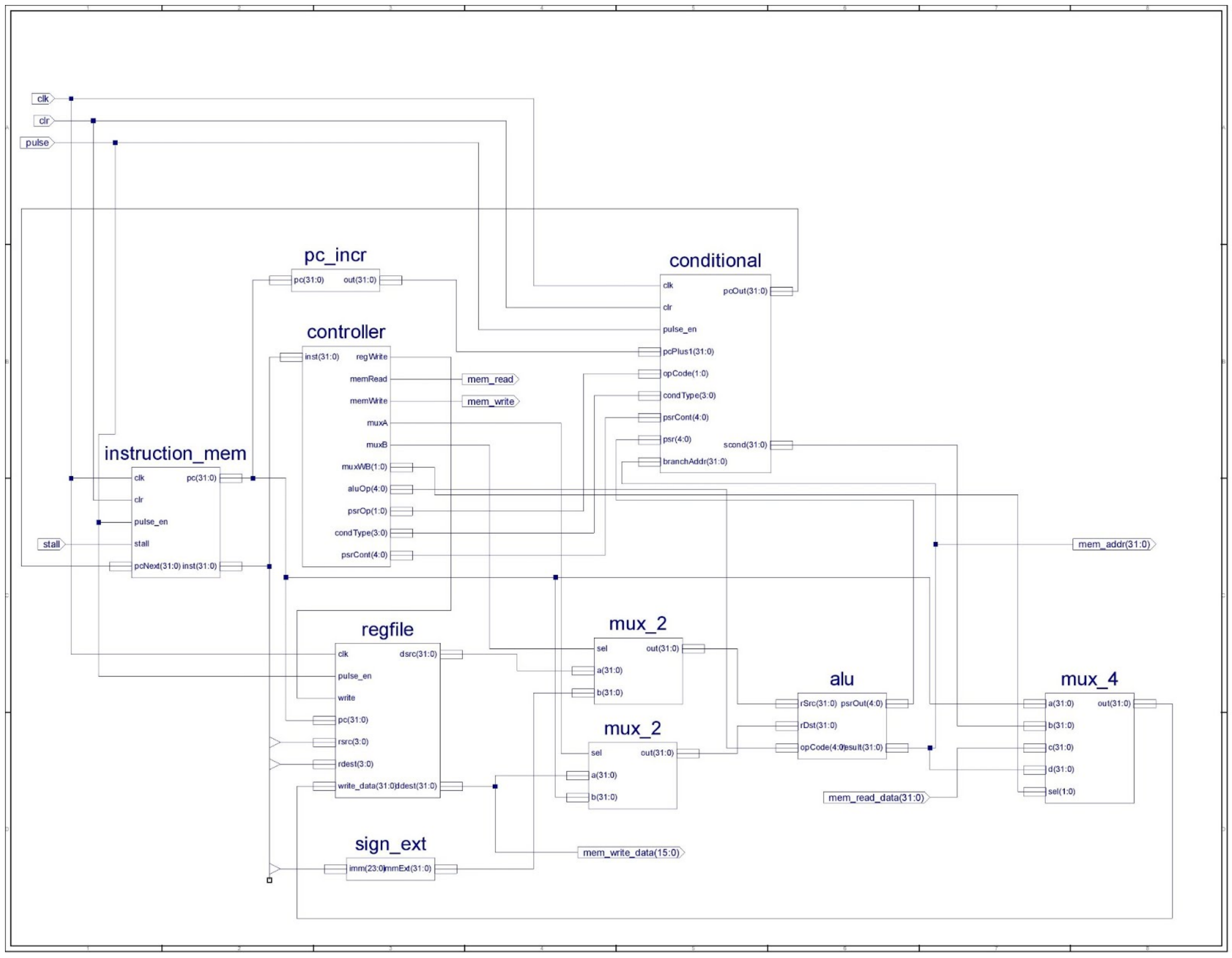
High Level Diagram:



Verilog Top Schematic:



Verilog Processor Schematic:



Hardware

1. Open /System/System.xise in Xilinx ISE version 14.3.
2. Generate a .bit file.
    a. Note that the instruction memory is configured to look at /Application/app.dat for memory initialization data.
    b. Note that the Map switch "-convert_bram8" must be set (see http://www.xilinx.com/support/answers/39999.htm).
3. Connect the FPGA microUSB programming cable and the FPGA microUSB UART cable to the computer, and turn on the FPGA.
4. Connect the FPGA VGA port to a monitor.
5. Open Digilent Adept program and select the Nexys3 FPGA.
6. Write /Application/gui/splash_screen.bin to RAM at decimal offset 614400.
7. Write /Application/gui/gui.bin to RAM at decimal offset 921600.
8. Go to the "Config" tab and program /System/system.bit onto the FPGA.

Assembler

Note: The Assembler application is compiled to run on Windows 7 x64. MS Visual Studio may have to be installed (or at least the runtime frameworks that come with MS Visual Studio).

1. Open a text editor and write an assembly file that adheres to Assembler README.txt and save with a .cr16 file extension.
2. Launch the the Assembler from /Assembler/Assembler/bin/debug/Assembler.exe
3. Click File > Open and open the .cr16 assembly file.
4. The file will now be assembled.
    a. If no errors are encountered, the status bar at the bottom of the application will show “Success!” and the resulting hex instructions will be visible alongside the assembly instructions.
    b. If there are errors, the application will show as many instructions as it could get through, and a pop-up will notify the user of the error. Errors should be corrected in the original .cr16 file, and from within the Assembler click File > Rebuild.
5. Once the assembly file is successfully assembled, click File > Save to write the assembled instructions to a .dat (or .coe) file that can be used to initialize FPGA block RAM.

Application

Note: The MainController application is compiled to run on Windows 7 x64. MS Visual Studio may have to be installed (or at least the runtime frameworks that come with MS Visual Studio).

1. Refer to the Assembler section of this User Guide to Assemble /Application/main_app.c16 and save it to /Application/app.dat.
2. Plug the FPGA into the computer, so that the UART Com. Port can be recognized by Windows.
3. Ensure there is a webcam connected to the computer.
4. Then launch the MainController helper application from /Application/MainController/MainController/bin/Debug/MainController.exe.
5. Within MainController, set the Com. Port to the one that the FPGA UART is connected to.
6. Click “Reset” and the user should see a preview of the image that the FPGA will take, streaming from the webcam. At this point, webcam settings can be adjusted by clicking “Options” but this is not necessary.
7. Refer to the Hardware section of this User Guide to configure the hardware.
8. The application will start by drawing a splash screen and waiting for the user click the middle button (C9:BTND) on the FPGA. At this point the image will begin coming in over UART from the webcam on the computer to the FPGA, overwriting the splash screen one line at a time.
9. Once the full image is loaded, the user may cycle through a series of image effect by pressing left and right buttons (C4:BTNL and D9:BTNR).
10. Once the desired effect is chosen, the user can apply it by pressing the center button again.
11. Send the image back to the computer by pressing the center button again. A message will be displayed in MainController when the JPEG has been generated.
12. On the FPGA, to go back to the splash screen, press the center button a last time. Now both the FPGA and MainController are ready to take another picture.