ECE 385

EXPERIMENT #7

VGA Text Mode Controller with Avalon-MM Interface

I. OBJECTIVE

In this lab, you will create a simplified text mode graphics controller which is connected to the Avalon memory-mapped bus and supports 80 column text mode through the VGA output. For the first week, you will create a simplified version of the monochrome graphics adapter, supporting only black and white text. In the second week, you will redesign and extend the graphics controller from week 1 around on-chip memory, enabling a much larger video memory (VRAM) space. This will allow you to draw different

II. INTRODUCTION

An Intellectual Property (IP) core is a reusable hardware design unit that is the intellectual property of a party. It can come in many different forms, ranging from hardware description language (HDL) to transistor layouts. An IP core in the hardware world is analogous to a library in the software language. The IP cores are often designed to gear towards specific functionalities rather than complete hardware systems. An IP core is usually specified through its inputs and outputs, with emphasis on the various design specs and performance evaluations such as the ones we have introduced in Experiment 4.

Users of an IP core usually obtain their license from the owner of the IP core to incorporate the IP core into their own design for extended functionality. As a notable example, manufacturers of ARM-based processors obtain their license from ARM Holdings to design their own processors. The owner of the ARM IP core does not manufacture its own processors, but instead focuses on the development of the ARM architecture itself.

In this lab, you will create a simplified text mode graphics controller IP core which is connected to the Avalon memory-mapped bus and supports 80 column text mode through the VGA output. This is functionally very similar to the original IBM monochrome graphics adapter (MGA) which was included with the IBM PC 5150 from 1981.

For the second part of this lab, you will extend the monochrome text display which you designed in the first week to support color text. This will require the use of additional video memory, so you will likely have to rethink your graphics controller around the limitations of on-

chip memory, rather than work directly from FPGA registers. In addition, you will have to modify the provided device driver to add proper color support.

Please read INTRODUCTION TO THE AVALON AND VGA (IAMM) as well as the VGA SPRITE DRAWING TUTORIAL (IVGA).

III. PRE-LAB

The recommended approach to this lab is to first read the INTRODUCTION TO THE AVALON AND VGA (IAMM), create the Avalon 'part' which will define your IP. As part of this step, you should also fill in the code to read and write the registers from Nios II through the Avalon bus. This portion may be independently tested through the provided test routines, irrespective of whether the VGA drawing portion is working. After the registers have been tested, you should read the VGA SPRITE DRAWING TUTORIAL (IVGA) and understand how to draw the sprites (in your case, the sprites will simply be the text glyphs in an 80x30 grid). This portion of your design will need to use the VGA controller from the previous lab, as you will need to use the DrawX and DrawY signals to determine where the 'virtual electron gun' is on screen. You will use this positional information as well as the contents of the video memory (VRAM) and the font ROM (font_rom) to determine which glyphs to draw and the bitmaps for each glyph.

IV. LAB

Follow the Lab 7 demo information on the course website.

V. POST-LAB

1.) Refer to the Design Resources and Statistics in IQT.29-31 and complete the following design statistics table.

LUT	
DSP	
Memory (BRAM)	
Flip-Flop	
Frequency	

Static Power	
Dynamic Power	
Total Power	

Document any problems you encountered and your solutions to them and write a short conclusion. Before you leave from your lab session submit your latest project code including both the .sv files and the software code to your TA. TAs are under no obligation to accept late code, code that does not compile (unless you got 0 demo points) or code files that are intermixed with other project files.

VI. REPORT

Write a report, you may follow the provided outline below, or make sure your own report outline includes at least the items enumerated below.

1. Introduction
    a. Briefly summarize the operation of the VGA interface, what are we trying to accomplish with this design?
    b. You should address how the design you created builds on top of the basic one provided for Lab 6.2.
2. Written Description of Lab 7 System
    a. Week 1 (Monochrome Text Display)
        i. Written Description of the entire Lab 7 system
        ii. Describe at a high level your VGA Text Mode controller IP
        iii. Describe the logic used to read and write your VGA registers
        iv. Describe the algorithm used to draw the text characters from the VRAM and font ROM (specifically, describe the equations required to generate the correct addresses to index into the VRAM as well as the font ROM).
        v. Describe your implementation of the inverse color bit, as well as the implementation of the control register.
    b. Week 2 (Color Text Display)
        i. Describe the hardware changes you had to make to support the use of multi-color text. At the minimum you must describe:
            1. Modification of register-based VRAM to on-chip memory-based VRAM. How did your design share the limited on-chip memory ports?

2. Corresponding modifications to the Platform Designer IP (e.g. Part Editor).
3. Modified sprite drawing algorithm with the updated indexing equations from on-screen pixels to VRAM.
4. Additional modifications necessary to support multicolored text.
5. Additional hardware/code to draw paletted colors

3. Block Diagram
    a. This diagram should represent the placement of all your modules in the top level. Please only include the top-level diagram and not the RTL view of every module.
    b. Note that depending on your layout of the registers inside your main module, the Quartus view may be illegible, in which case you should draw a block diagram using software. You may start from the provided materials (e.g. in IAMM), but you should fill in the specific signals between the modules and the inside subcomponents within each module.
    c. You should have block diagrams for both the Week 1 and Week 2 portions, a good setup is to show the common components (e.g. the SoC setup) first and then show diagrams for both the Week 1 and Week 2 VGA controller component.
    d. If your design has a state machine, you should include a State Diagram as well.
4. Module Descriptions
    a. A guide on how to do this was shown in the Lab 6 report outline. Do not forget to describe the Platform Designer generated file for your Nios II system! When describing the generated file, you should describe the PIO blocks added beyond those just needed to make the NIOS system run (i.e. the ones needed to communicate with the USB chip and other components). The Platform Designer view of the Nios II system is helpful here.
5. Document the Design Resources and Statistics from the lab manual. Each week's design should have different design statistics, and you should briefly discuss the difference between using on-chip memory for VRAM and registers. Which design is more efficient, what are the tradeoffs?
6. Conclusion
    a. Discuss functionality of your design. If parts of your design didn't work, discuss what could be done to fix it.
    b. What are some potential extensions of this design, what did you learn in this lab that might be useful for your Final Project?
    c. Was there anything ambiguous, incorrect, or unnecessarily difficult in the lab manual or given materials which can be improved for next semester? You can also specify what we did right, so it doesn't get changed.