ECE 385

EXPERIMENT #7

HDMI Text Mode Controller with AXI4 Interface

I. OBJECTIVE

In this lab, you will create a simplified text mode graphics controller which is connected to the AXI4 memory-mapped bus and supports 80 column text mode through the HDMI 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 text characters in different colors.

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 AXI4 memory-mapped bus and supports 80 column text mode through the HDMI 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.

III. PRE-LAB

The recommended approach to this lab is to first read the INTRODUCTION TO THE AXI4 AND HDMI (IAMM), create the AXI4 'part' which will define your IP. In addition, you should set up the HDMI IP with the components from the previous lab to draw at least the gradient from the previous lab through HDMI. This will prevent the HDMI output signals from being optimized out and allow you to generate the bitstream for the hardware. In addition, you should make sure you fill in enough of the testbench so that you can simulate an image.

Once you can simulate and display a static image and the IP can be placed into the hardware design, you should also fill in the code to read and write the registers from MicroBlaze through the AXI4 bus. Understanding the provided testbench will also be helpful for this portion. This portion may be independently tested through the provided test routines, irrespective of whether the HDMI drawing portion is working. Finally, after the readback test is correct in simulation and the checksum test works on the hardware, you should implement the drawing of the character display based on the contents of the registers. For this portion of your design, you will need to use the HDMI and VGA controllers 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

Please read INTRODUCTION TO THE AXI4 AND HDMI (IAXI). This document contains the necessary information to set up the IP, as well as set up simulations within the IP's "Edit Project". A basic test bench is provided, which you must modify to simulate an image. Once you are able to simulate the image being drawn, you will be able to more easily debug without going through a long build process for each change. **It is not recommended to approach the main hardware design until your simulation is able to properly output a static image**.

Once you are able to simulate, proceed with the construction of the IP block until you are able to generate an image similar to Figure 21 in IAXI. Once you have completed this step, attempt the hardware implementation and run the corresponding test routine in software.

V. POST-LAB

1.) Refer to the Design Resources and Statistics in the IVT tutorial 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 your lab session submit **only the zipped IP directory for the lab week (week 1 or week 2). Do not include your project, it doesn't contain any unique code.**

VI. REPORT

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

1. Introduction
    a. Briefly summarize the operation of the HDMI 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.
    c. In Lab 6.2, we had a different method for hardware->software communication (e.g., the keycode). Describe some advantages/disadvantages of the IP approach in Lab 7 compared to the approach in 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 HDMI Text Mode controller IP.
iii. Describe the logic used to read and write your HDMI AXI 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 color control register.
vi. Describe how you implemented the frame counter and other registers. Specifically discuss in detail how the frame counter works.

b. Week 2 (Color Text Display). Describe the hardware changes you had to make to support the use of multi-color text. At the minimum you must describe:
   i. Modification of register-based VRAM to on-chip memory-based VRAM. How did your design share the limited on-chip memory ports?
   ii. Corresponding modifications to the IP Editor.
   iii. Modified text drawing algorithm with the updated indexing equations from on-screen pixels to VRAM.
   iv. Any additional modifications which were necessary to support multicolored text.
   v. Additional hardware/code to draw paletted colors. Describe your C code for setting palette values.
   vi. If the implementation of the frame counter and other synchronization registers is different from Week 1, describe those differences here. Otherwise, you can simply write that it's handled in the same way.

3. Block Diagram
   a. This diagram should represent the placement of all your modules in the top level.
   b. Internal block diagram of the IP you designed for each week.
      i. Note that depending on your layout of the registers inside your main module, the Vivado schematic view may be illegible, in which case you should draw a block diagram using software. You may start with the materials provided (e.g., in IAXI), but you should fill in the specific signals between the modules and the inside subcomponents within each module.
      ii. 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 HDMI controller components.

iii. 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 Black Design file for your MicroBlaze system.
    b. Describe the internals for each week's version of the HDMI controller IP. Note that shared components may have a single module description.
    c. If any modules needed to be changed between Week 1 and Week 2, describe the changes here as well.
5. Simulation Waveforms and Images
    a. Annotated simulation of AXI write transaction (you may use either Week 1 or Week 2 design).
    b. Annotation simulation of AXI read transaction (you may use either Week 1 or Week 2 design).
    c. Detailed description / commented code of the axi_read () task you filled out within provided testbench.
    d. Simulated image from Week 1 testbench (see IAMM document, this should not have the red text so it should be generated by you).
6. Additional Testbench and Simulation Image
    a. Simulated image from Week 2, this text shows the following string "netid1 and netid2 completed ECE 385!". Fill in your and your partner's NetIDs. The NetIDs should display in blue (0, 0, F) while "ECE 385!" should display in orange (F, A, 0). The background and the rest of the text may be any color of your choosing provided they are distinguishable / legible (e.g. white and black are acceptable). You may add additional exclamation marks (or omit the exclamation mark altogether) so that the resulting string is a multiple of 2 characters so that you do not need to modify the provided axi_write() task to support write strobe.
    b. The commented portion of the modified test bench which calls appropriate AXI writes that display the simulated image as in part a.
7. Photo of Week 2 design showing correct NetIDs of the students in the screensaver test routine. This can either be a photo of a monitor/TV directly or a screenshot from a capture card.
8. 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. You should include both sets of design statistics. Which design is more efficient, what are the tradeoffs?
9. 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.