Digital Design and Computer Architecture

SDRAM and VGA Integration Lab

By: Kent Jones

For this in class exercise, you will become more familiar with the SDRAM module on the XSA-3S1000 FPGA board. The goal of this lab to display colors read from the SDRAM memory onto the VGA display. **We will work on this lab together in class.**

Grading

This lab will not be graded, however, if you learn the material well, it could help you design a video output for your final project processor, depending on how much time you are willing to devote to integrating the knowledge you learn in this lab with your processor project.

Contents

Introduction

To obtain a basic understanding of the SDRAM module input, we will work with an example project provided by XESS. We will then combine the SDRAM module with our VGA module to display the contents of the SDRAM memory on the VGA monitor.

Goals

- Work to increase understanding of how the SDRAM module works.
- Work to increase understanding of how the more complex VGA generator module works.
- Implement and test the XSA SDRAM controller example.
- Test the XSA SDRAM module to the VGA module to display contents of the XSA's memory
- Modify the algorithm that generates data in memory so that the VGA generator displays a different test pattern

PART 1: Understanding Single Port SDRAM Controller

This lab will begin by investigating how the SDRAM for the FPGA board is structured.

1. First, we will navigate to the XESS website and read about the single port SDRAM controller module. You can read about this module here: http://www.xess.com/static/media/appnotes/an-071205-xsasdramcntl.pdf

2. Read page 1 of the document. How many rows and columns does the XSA-3S1000 SDRAM memory have?

   ROWS: __________________ COLS: ______________________

3. What are the purposes of the **FREQ** and **CLK_DIV** generic parameters? You will see these again later!

   ____

4. Look at **Figure 1** on **page 6** of the document. What **signal** carries the address of the memory location to be read/written from the FPGA host logic to the XSASDRAMCntl?

   ____

   What **signal** carries the actual data to the XSASDRAMCntl from the FPGA host logic?

   ____

5. Look at **Figures 2 and 3** on **pages** 7 **and 8**. The description of these signals are on pages 2 and 3. These are the timing diagrams for the non-pipelined read and write operations of the single port controller. Once you have set the address you want to read using **`hAddr`**, and asserted the **`rd/wr`** signal, what signal(s) from the controller will tell the FPGA host logic that the data has been successfully retrieved from or written to the sdram?

   ____

6. Look at **Figures 4 and 5** on **pages 9 and 10**. What are the major differences between the **pipelined read/write operations** and the **non-pipelined read/write operations**?

   ____
   ____
   ____

PART 2: Connecting the VGA to the Simple Single Port RAM

1. Navigate to http://www.xess.com/projects/an-101204-vgagen-project/ and download the an-101204-vgagen Project from XESS to your desktop. **NOTE: Be sure and download the associated pdf at the same time!** http://www.xess.com/static/media/appnotes/an-101204-vgagen.pdf

2. This project does not use a simple VGA generator like we used when we were learning how the VGA output works! This uses a more complex VGA generator that implements a pixel buffer. You can read about this generator in the pdf file you downloaded in step 1.

3. Make a **VGASDRAM** folder.

4. Extract the **an-101204-vgagen** folder and place it inside your **VGASDRAM** folder.

5. Navigate to **an-101204-vgagen\3S1000\test_vga** and double-click on the **test_vga.npl** file. Select **Migrate Only** to migrate the hardware design to the new ISE project format.

6. Synthesize the project, implement the design. (Ignore the warnings, not a good habit in general, but it works in this case because the project was updated.)

7. Right click on **Generate Programming File** and select **process properties**, set all the configuration options so that all the **pull-ups** and **pull-downs** are changed to **Float**.

8. Generate the programming (.bit) file. Run GXSLOAD, **drag and drop** the newly generated **test_vga.bit** file to the **FPGA/CPLD** field of GXSLOAD and **ALSO AT THE SAME TIME** drag and drop the **image.xes** file (located just inside the an-101204-vgagen folder) to the **RAM** field of GXSLOAD. Load both files to the FPGA.

9. If you run the VGA on an LCD panel, you may have to adjust the horizontal position to see the whole image correctly.

10. You should see an image form on the VGA monitor. What is this an image of?

    ____

PART 3: Understanding the Dual Port SDRAM Controller

1. Now, navigate to the XESS website and read about the Dualport Module for the SDRAM controller here: http://www.xess.com/static/media/appnotes/an-071205-dualport.pdf

2. The main signals for the **Port to SDRAM Controller** are listed on the first page of this document. Give a simple description of the purpose of each signal in the table below (**You will need to refer back to the single port SDRAM controller documentation to learn more details of the purpose of each signal**):

Port to SDRAM Controller	**Description of Signal**
`rst`	
`rd`	
`wr`	
`earlyOpBegun`	
`opBegun`	
`rdPending`	
`done`	
`rdDone`	
`hAddr`	
`hDIn`	
`hDOut`	
`status`	

3. What is the purpose of the **PORT_TIME_SLOTS generic parameter**?

____

____

4. Read the descriptions of each of the major files in the project and also read about the LED segments and how they are activated in response to the tests. **How will you know if a test fails or not?**

PART 4: Running the Sample Dual Port SDRAM Controller Example

Locate the sample project files found at http://www.xess.com/projects/dualporttst-1_1-project/ . This holds the design files for the SDRAM controller project. Read the DESIGN FILES and USING THE DESIGN EXAMPLE sections under the description of the project on this page.

1. Download the project file to the VGASDRAM folder you created in your CS401 folder on CS1. Extract the project folder from the .zip file into this VGASDRAM folder.

2. Browse in windows explorer to your folder **VGASDRAM\dualporttst-1_1\dualporttst\XSA\3S1000\test_dualport**

3. Load the **test_dualport.bit** file into the FPGA board using **GXSTEST**. NOTE: You don't have to build it yet; it's already been built. We will rebuild it in **Part 5**.

4. Run the test by holding down pushbutton **SW2** on the **FPGA** board. What happens when you hold this button down and why? **HINT**: read "USING THE DESIGN EXAMPLE" found here: http://www.xess.com/projects/dualporttst-1_1-project/

PART 5: Building the Sample Dual Port SDRAM Controller Example

1. Browse to **VGASDRAM \dualporttst-1_1\dualporttst\XSA\3S1000\test_dualport**

2. Double – click on the **test_dualport.npl** file to open the project in Xilinx ISE. When the dialog box appears, migrate the project to the new version of the ISE.

3. **Synthesize the project and implement the design. You will run into an error…**

4. **Comment out lines 8 and 9 of the test_dualport.ucf file. The Digital Clock Managers do not need to be manually placed.**

5. Implement the design using the ISE interface.

6. Before generating the **.bit** programming file, don't forget to set all the **pull-ups** and **pull-downs** to **Float** by right-clicking on **Generate Programming File**, selecting **Process Properties** followed by **Configuration Options**.

7. Now, delete the existing **.bit** file and re-generate it. Use GXSLOAD to load the .bit file generated by the project to the FPGA.

8. Hold the SW2 pushbutton down, and the lights should cycle through the SDRAM test.

PART 6: Understanding the Dual Port Example

1. Examine **test_dualport.vhd**. What is this hardware description doing?

2. Examine the **test_duaplport_core.vhd**. What is this hardware description doing?

PART 7: Understanding the Dual Port VGA / SDRAM Example

1. Copy the **DUAL_PORT_SDRAM_AND_VGA_V3** folder from \\cs1\CS_ClassData\401_Computer_Architecture\ to your **VGASDRAM** folder.

2. Browse to the following subfolder **after you have it copied** to your **VGASDRAM** folder:

   VGASDRAM\DUAL_PORT_SDRAM_AND_VGA_V3\dualporttst\XSA\3S1000\test_dualport

3. Open the project (double-click on **test_dualport.xise**)

4. Delete any **.bit** files in this same folder. Synthesize, implement, generate the programming file, and finally GXSLOAD the generated bit file into the XSA board.

5. Connect to a VGA monitor. You should see a red and blue test pattern appearing on the screen.

6. Look at lines 267 through 312 of test_dualport_core.vhd. This is the hardware process (starting on line 269) that generates the test pattern. (Remember that in a process we can use := assignments which are immediate rather than parallel and delayed like signal assignments <=) NOTE What is happening on line 266?

   ____

7. Now, look at line 379. What does the **`hDIn0`** signal connect to? What port are these signals connected to on the Dual Port SDRAM controller?

   ____

8. Let's look at the process that outputs color to the VGA pins. Look at lines 332 to 355. This is the top level of the vga_generator. What is happening at line 346? Note that **`hDout1`** is connected to the **pixel_data_in** of the vga generator.

9. Where does **`hDout1`** come from? Look at lines 384 to 395. What port are these signals connected to on the **Dual Port SDRAM controller**?

   ____

PART 8: Modifying the Dual Port VGA / SDRAM Example

1. **Now, it's time to make your own test pattern**.

2. Copy the Dual Port Example that you built in the previous part. Rename the folder **LAB_5_Test_Pattern**. Change the VHDL code that generates colors that are placed in memory. Make your own test pattern.

3. Think about structural VHDL. What would be required to replace the test pattern generator with output from your microprocessor?

Appendix: Tips for Building VGA / SDRAM / Microprocessor Projects

You really have two options to integrate VGA with your microprocessor.

Option 1: You could start with a simple VGA generator project that you built last semester and integrate it directly with a microprocessor.

Pros: Easier to understand the VGA generator, you could have very simple output to the VGA that is controlled by your program (e.g. an output port on your processor simply changes the color of a box that the VGA is displaying)

Cons: Much, much, more challenging to create a text generator or other types of output to the VGA.

Option 2: Start with the project from part 7 of this document.

Replace the process that writes a test pattern to memory with a module that integrates your microprocessor. Thus, instead of a hardcoded VHDL process writing a pattern to the SDRAM, your microprocessor will need a program that writes to the memory in the SDRAM that the VGA reads and displays.

Pros: Much more flexible output to the VGA, much easier (once you get it working(to create bit-mapped graphics and text output, etc.

Cons: Much more challenging to integrate with your microprocessor, but, if you understand VHDL structural code, you should be able to keep the microprocessor hardware separated from the VGA hardware.