Lab 5: Parallel Color Graphics, RAMs, and ROMs

Concepts and Background

The display mounted to the expansion board uses a parallel data interface to load the pixel data that creates the image on the display. Unlike the ADC and DAC, now 24 bits of data are transferred each clock cycle in reference to two additional control signals, synchronous to a clock. Those two control signals are for horizontal and vertical synchronization, allowing the display driver to correctly route and store each active pixel's data.

Procedural frame generation is a graphics rendering technique for low-performance or memory-limited image rendering. It takes advantage of the fact that the display interface isn't used to transmit the whole image at once, but instead transmits the pixel data sequentially. Rather than storing the data (color) value for each pixel in the image, there is a function (or procedure, to follow the terminology) that computes the color value for each pixel when given a pixel's coordinates. This way, a smaller amount of memory is required to store all the information required to create the full image that will be sent to the display.

To store the relevant data, a procedural system will use a combination of Random Access Memories (RAM) and Read Only Memories (ROM) to hold the dynamic and fixed information, respectively. A truly random-access RAM enables data to be read or written at arbitrary addresses each access cycle. The Cyclone V FPGA provides embedded block memories that provide two independent 'ports' to a single block, enabling two systems to access the same memory efficiently and even with different clocks. These blocks can also have their contents set on device initialization to allow for a known initial state or for the creation of ROM blocks. The creation of the ROM is simply the same memory with a single read-only port.

Implementation Requirements

This lab is an implementation of a fixed-grid, editable character display. The details of the input system are left to the implementer, but must be documented. Use of buttons, switches, and the motor encoder are all the easily available interface components. The video output will be displayed on the expansion board's 24-bit color LCD.

The input controls a cursor in the cell array of characters that indicates where user input will be placed and enables the user to enter data into the desired cell. Since selection of both an X and a Y coordinate are required, the controls should provide for such input. Possible methods include setting the coordinate being manipulated with a switch and incrementing/decrementing the position counter using buttons or the motor encoder. The selection should limit the selection of cells only on the screen. The character setting should use 7 switches to define the character code and a button to activate the write to the cell selected.

The display is sent a procedurally generated graphical representation of the contents of the cell array. In addition to displaying the array of characters, the cell that is being edited should be highlighted in an obvious fashion. The array should be implemented as outlined in the design questions. One such method is swapping the background and foreground colors used to display the 1-bit data from the font ROM. The foreground and background colors chosen should be high contrast (e.g. bright blue on black, dark red on white) such that the characters are discernible on the display. Further use of color is permitted so long as the characters remain discernible.

Demonstration Goals

To establish that the group has read and properly understands the specification, the group must demonstrate that all listed features and requirements are implemented correctly. Please prepare your demonstration to be able to quantitatively show each of the following requirements.

- Procedurally generated, color graphics displayed on screen
- Fixed grid of editable data cells is visible and fills the screen as analyzed in the questions
- Intuitive user interface for selecting which cell to edit and changing the data in the cell
- Cells contain characters that are stored in the font ROM
- Minor graphical glitches or inconsistencies are permitted so long as functionality is not compromised

Code Organization Requirements

- Clear separation of clocking using a dual-port RAM between clock domains or entirety of design uses 9MHz display clock; all flip flops in each domain use the same clock
- Modularization of design: Verilog modules for each unit of functionality of the design
- Organized files and directories for Verilog modules and configured MegaFunctions
- Top-level module contains only wires, buffers/inverters, and module instances
- Assignment of meaningful names to the GPIO interface signals from the expansion board
- Correctly formatted module headers and organized code
- Project submitted on BlackBoard, meeting the submission policy requirements

Design Recommendations/Hints

- Read Altera's *Dual Clock Synchronous RAM* design example. The list of all examples is linked in the Verilog and HDL Design References section of the Course Documents. This is the exact type of RAM that you will need to store the character array.
- The display backlight requires the 12V power supply. No image will appear on the display without the backlight turned on.
- Start by completing the timing-critical *video_position_sync* module and testing it with the display using the video demonstration project.
- Using sub-vector selection on the x_pos and y_pos signals to create at least parts of the addresses for the RAM and ROM.

Design Questions

For all questions, show the supporting calculations used to determine the answer as required. Answers given without showing work will not receive full credit, even if the answer is correct.

1. With a 9MHz display clock, calculate the frequencies of the horizontal and vertical synchronization signals from the display's timing diagram.
2. What size character does the supplied font ROM hold? This will be the size of a "character cell."
3. What is the display's resolution (visible pixels)?
4. Using the answers to the above two questions, how many complete character cells can be displayed in a rectangular grid on the display? How many cells are in each row? Each column?
5. How many visible pixels on the display do character cells not cover? These would be pixels that cannot display a complete character along the edge of the display.

6. Using a grid of power-of-two side lengths, what size grid is needed to hold the data for this screen-filling array of stored characters? How many cells are unused because of the power-of-two addressing constraint? For example, a 3x6 grid of characters requires a 4x8 grid of RAM storage for ease of addressing.
7. How should a user manipulate your system? Explain the interface you implemented for data entry into your design.

Report Requirements

- Complete answers to the above design questions.
- Discuss how the two data flow systems in your design work: the user input to the character storage RAM and the continuous procedural graphics generation using the RAM and ROM.
- Discuss the functionality of each module used in your design, including the modules' different interfaces. Your signal names should make sense relative to their function and correspond to the signal names used in the block diagram.
- Include a block diagram of the connections between the modules in your design and their connections to external signals through the top-level.
- Include oscilloscope captures of the relationship between the display clock, horizontal sync, and vertical sync. This will require multiple timescales due to the significant differences in frequency. Measure the frequency of each signal using the oscilloscope and compare to the frequencies you calculated in the design questions.
- Include SignalTap captures of a complete horizontal line containing data in the active area of the screen and a blacked-out line in the vertical sync region. The captures *must* have at least hsync, vsync, and one 8-bit color bus. Provide a caption/description explaining the captures.
- Include a SignalTap capture of how your design accesses the font ROM. The capture should include the x_pos and y_pos signals, the read address to the ROM, the data output from the ROM, and the sub-word address for the ROM output. Provide a caption/description explaining how the signals are related.
- Summarize how each group member contributed to the completion of this lab.

Grading Distribution

- Demonstration 30%
- Code Organization 10%
- Design Questions 20%
- Oscilloscope Captures 5%
- SignalTap Captures and Captions 15%
- Report Discussion 30%