EE 3610 Digital Systems
Lab 5

Title: Character Generator.

Objective: The student will gain experience using block memory lookup tables and working with video signals.

Equipment: Spartan 3E Starter Board
VGA monitor that accepts the 1024x768 XGA format.

Background: In this lab, you are to design a component that converts ASCII codes in a (yet to be implemented) character memory and displays them on the VGA monitor.

A “dumb” terminal divides the visible area of the screen into rows and columns, then it displays one character in each cell of the grid. For example, if the size of one character is 11x16 (11 pixels wide and 16 pixels high), there would be 48 rows and 93 columns on a 1024x768 screen (one pixel column would be unused). The ASCII codes for these characters are stored in a “character memory”.

As each line of VGA data is transmitted, the character generator must fetch the characters for that line from character memory, then based on that character and the line number within the row, it must look up the row of pixels in the font memory that should be displayed. This row of pixels is then loaded into a shift register and shifted out at the pixel clock frequency.

The font memory is addressed by combining the ASCII code and the line number within the row. The font memory then outputs the pixels associated with the given line of the given character. For example, if the font size is 11x16 pixels and line 6 of the pixels in the letter ‘A’ (ASCII 41_{16}) is 00010010000, then the contents of font memory at location 416 might be 090_{16}.

With Xilinx FPGAs, memory such as the font and character memories can be implemented either as distributed memory (which uses logic blocks) or as block memory (which uses dedicated memory blocks). ISE infers one type memory or the other depending on how you access it. For example, if the output of the memory is piped through a D-register, ISE infers block memory:

```
process (clk)
begin
  if clk='1' and clk'event then
    the_data <= the_memory(to_integer(the_addr));
  end if;
end process;
```

See also Section 6.6.1 of the text. Both the font and character memories should be implemented in block memory.

Preparation: Write the title and a short description of this lab in your lab book. Make sure the page is numbered and make an entry in the table of contents for this lab.

The schematic for this lab is identical to that of Lab 4, so you may simply refer to that schematic in your lab book.

Choose a font size and design or acquire the pixels for that font. You may use any font size you wish, but you may also use the 11x16 font that has been provided on the course web site and is already converted to VHDL.

Design a character generator module that interfaces to your VGA module on the one hand and to the (yet to be implemented) character memory on the other. The interface to your VGA module should include at a minimum the horizontal start signal (asserted one cycle before pixel data starts) and the current line number. You may also include vertical sync, horizontal sync and/or blanking if that simplifies your design.

The interface to the character memory consists of an address output and a data input. The number of bits in the address depends on the number of rows and columns on your screen. For example, if you have 48 rows and 93 columns, the number of address bits would be 13 (6 to specify the row and 7 to specify the column). The data input need only accommodate ACSII codes in the range 0 to $7F_{16}$.

Of course, your module must input the reset and 65MHz clock, and it must generate a 1-bit pixel data output.

Note that the output of the font and character memories must be piped through a D-register so that ISE will infer block memory. This means that all accesses to memory must occur at least one cycle before you need the data.

Also note, in Lab 7, we will modify this module to generate a cursor output given a current cursor location (row and column). This output is active whenever the character at the cursor location is being displayed. You may include this feature now if you wish, but it is not required.

Write a test bench for your character generator module. Your test bench should provide the ASCII data that would normally come from the character memory. Make sure to display at least 2 different characters. Verify that both the address to character memory and the pixel data out are correct. Simulate at least two horizontal lines (make sure you have distinct pixel data). Zoom in to show the first two characters on the first line (e.g. the first 22 pixels), print the waveform and affix it to your lab book. Repeat this procedure for the second line.

Modify the top level module from Lab 4 (call it lab5.vhd) to include your character generator module. Drive red green and blue from the pixel data out (except, of course, during the blanking intervals). Either create a character memory and initialize it with interesting characters or download the character memory module on the course website. Connect the character memory to the character generator and synthesize your design to verify that there are no syntax errors.

Bring your lab notebook and the Spartan board, above, to your lab period.

Set up: Connect the USB cable, VGA monitor and power supply to the Spartan board. Turn on power to the monitor and the Spartan board.

Procedure: Download your code. Verify that the characters you initialized in character memory appear on the monitor in the right places. Demonstrate your system to the lab instructor.

Affix the final copies of your character generator and top-level modules (in VHDL) to your lab book.

Conclusions: In the conclusion section, write a short summary of what you did, what you learned, and what could be done better.