EE 3610 Digital Systems
Lab 6

Title: Character Memory.

Objective: The student will gain experience manipulating block memory using a controller.

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

Background: In this lab, you are to design a component that manages character memory. The component must support the following 3 operations:

1. Write a character to a specific location.
2. Clear the screen (write all memory locations to blank)
3. Scroll (move row n to row n-1 and blank the bottom row, n>0)

The block memory on the Spartan 3E is dual ported, which means that two different processes can be accessing memory at once. One port will be used to output data to the VGA. The other port will be used to manipulate the data in the character memory.

Writing a character to a specific location is straightforward, but clearing the screen and scrolling are a little more challenging. You will need to implement a controller to do it. You will want an idle state (where writing to a specific location can occur). When a "clear screen" signal occurs, your controller must generate the address of every character on the screen (one per cycle) and cause a blank (20_{16}) to be written to each.

The scroll operation is even more tricky. When a "scroll" signal occurs, your controller must copy each row to the row above it and blank (write 20_{16} to) the last row. One approach to this problem is to observe that block memory can be read and written simultaneously, in other words, you can read the old value at the same time you write a new one. You can use this feature to scroll memory in an efficient way. Simply generate memory addresses from bottom to top (one column at a time) and either write a blank (for characters on the bottom row) or whatever came out of the memory on the previous cycle (for all other rows). Note that this sequence of addresses visits every character location on the screen, so you can also use it (the sequence) to clear the screen.

After all the cells in memory have been updated (either by clearing or scrolling), your controller should return to the idle state.

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 practically identical to that of Lab 4, so you may simply refer to that schematic in your lab book.

Add inputs to the character memory module from Lab 5 to provide an interface for writing, clearing and scrolling. There should be three separate inputs to request each of these operations. You may assume these inputs never occur simultaneously and that any input that occurs while your module is busy (i.e. clearing the screen or scrolling) may be ignored. The character-write operation will need additional inputs (to the module) to specify the character to be written and the address in character memory (e.g. row and column) to write.

Remember that there are only 2 ports on the block memory. If you write your VHDL in any way that implies 3 or more ports, the current version of ISE will become stuck in an infinite loop. Use a single process statement for each memory port and use a multiplexer to select the appropriate memory address if necessary.

Write two test benches for your character memory module. Your first test bench should write at least one character to memory then clear the memory. The second test bench should write at least one character to memory then scroll the memory. Make sure all the addresses are generated properly and that the memory is being written to only when it should be. Affix a simulation showing the start (first few memory cycles) of the clear and scroll operations to your lab book.

Copy the top level module from lab 5 (call it lab6.vhd) and modify it to include the new ports in the character memory module. To test your memory module, add ports to lab6 for the three push buttons (btn_east, btn_west and btn_north) and add the lines below to the constraint file:

```
NET "BTN_EAST" LOC = "H13" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "BTN_NORTH" LOC = "V4" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "BTN_WEST" LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN ;
```

You will need to add the following signals to lab6.vhd:

```
constant DEBOUNCE: integer := 262143;
signal n_clear, n_scroll, n_write: integer range 0 to DEBOUNCE;
signal b_clear, b_scroll, b_write: std_logic;
signal p_clear, p_scroll, p_write: std_logic;
signal row: integer range 0 to 47;
signal col: integer range 0 to 93;
signal ch_row: std_logic_vector(5 downto 0);
signal ch_col: std_logic_vector(6 downto 0);
signal ch_data: std_logic_vector(6 downto 0);
```

Finally, the code on the next page can be used to clear, scroll and write characters to the screen using the pushbuttons. It debounces the buttons and generates a 1-cycle clear pulse (p_clear) when the East button is pressed and a 1-cycle scroll pulse (p_scroll) when the North button is pressed. It also monitors the West button and writes a line of ASCII data to the character memory when that button is pressed.

```
-- generate debounced signals for clear, scroll and write
  process(clk, reset)
  begin
    if reset = '1' then
      b_clear <= '1';
      b_scroll <= '1';
      b_write <= '1';
      n_clear <= 0;
      n_scroll <= 0;
      n_write <= 0;
    elsif rising_edge(clk) then
      if btn_east = b_clear then n_clear <= 0;
      elsif n_clear /= DEBOUNCE then n_clear <= n_clear+1;
      else n_clear <= 0; b_clear <= not b_clear;
      end if;
      if btn_north = b_scroll then n_scroll <= 0;
      elsif n_scroll /= DEBOUNCE then n_scroll <= n_scroll+1;
      else n_scroll <= 0; b_scroll <= not b_scroll;
      end if;
      if btn_west = b_write then n_write <= 0;
      elsif n_write /= DEBOUNCE then n_write <= n_write+1;
      else n_write <= 0; b_write <= not b_write;
      end if;
    end if;
  end process;

  -- connect these to your character memory
  p_clear  <= '1' when n_clear=DEBOUNCE and b_clear='0' else '0';
  p_scroll <= '1' when n_scroll=DEBOUNCE and b_scroll='0' else '0';

  -- write one line of characters when user presses west button
  process(clk, reset)
  begin
    if reset = '1' then
      row <= 0;
      col <= 93;
    elsif rising_edge(clk) then
      if n_write = DEBOUNCE and b_write = '0' then
        if row = 47 then row <= 0; else row <= row+1; end if;
        col <= 0;
      elsif col /= 93 then
        col <= col + 1;
      end if;
      if;
  end process;

  -- connect these to your character (screen) memory
  p_write <= '1' when col /= 93 else '0';
  ch_row <= std_logic_vector(to_unsigned(row,6));
  ch_col <= std_logic_vector(to_unsigned(col,7));
  ch_data <= std_logic_vector(to_unsigned(col+33,7));
```

Connect p_clear, p_scroll, p_write, ch_row, ch_col and ch_data to your character memory module, then 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 and run it. Using the West push button, insert several lines of ASCII characters into character memory and verify they are displayed on the monitor. Press the North button and verify the characters on the screen scroll up. Finally, press the East button and verify the screen is cleared. Demonstrate your system to the lab instructor.

Affix the final copy of your character memory module (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.