University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Science

EECS150, Spring 2010

Homework Assignment 7: Memories and Video

Due March 12^{th}, 2pm

Homework submissions must be made via the course SVN repository. Email submissions will not be accepted! Please format your homework as plain text, or PDF. You may use PNG for any necessary figures. Use a CAD program for diagrams (Microsoft Visio is installed on the machines in 125 Cory).

1. Your task is to implement a 32K x 32 memory. Assume you are given 32Kbit configurable memory blocks such as a block RAM in a FPGA. Each block can be configured as a 32K x 1bit memory or a 1K x 32bit memory. Which option would you choose as a building block for your 32K x 32bit memory and why?

2. (a) Write a parameterized memory in Verilog. It should be *simple dual port*. Write your module using behavioral verilog to infer the memory, as shown in lecture. The required parameters are the size (Width and Depth) and SynchronousRead.

parameter	**Description**
Width	Memory data width
Depth	Number of memory entries
SynchronousRead	if set to 1 then the memory is synchronous read; if set to 0, the memory is asynchronous read

   (b) Create a Xilinx ISE project and add your RAM module. Set the parameters to implement each of the following memories. For each, record the type of resource used, the percentage of this type of resource that is used up, and the memory configuration used. This information is near the bottom of the synthesis report.

      i. 32 x 1b memory with *asynchronous* read.
      ii. 4K x 16b memory with *asynchronous* read.
      iii. 32K x 32b memory with *synchronous* read.

3. We want to build a 32 x 32bit register file, with 2 read ports and 1 write port, on an FPGA. Refer to Virtex-5 FPGA User Guide (ug190.pdf) on the Documents page for this problem.

   (a) Build the register file from Block RAM, using as few resources as possible. Description of Virtex-5 Block RAM can be found starting at pg.109 of the User Guide. Draw your design and give a brief description, including what configuration is used.

   (b) Build the register file from Distributed RAM (LUT RAM), using as few resources as possible. The basic primitives are found on pg.207 of the User Guide. There may also be useful information within *Chapter 5: Configurable Logic Blocks (CLBs)*. Draw your design and give a brief description, including what configuration is used.

(c) Assume the rough area costs below:

Component	Area (μm^2)
LUT5	850
LUT6	1700
BRAM36	370000
BRAM18	185000
FlipFlop	100

i. Approximately how much area does it take for 36Kbits of storage using:
   A. Block RAM?
   B. LUT RAM?
   C. FlipFlops?

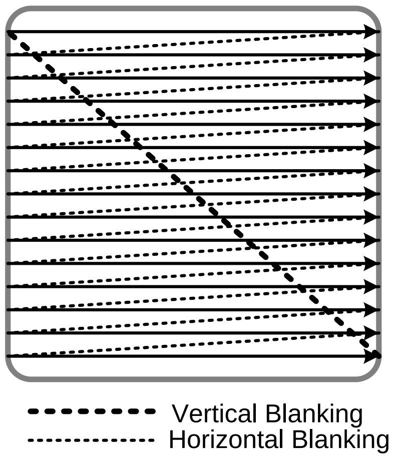
ii. How much area does each of your register file implementations require?
   A. Block RAM
   B. LUT RAM
   C. FlipFlops (estimate based on number of flipflops, as well LUTs for surrounding logic).

4. We are using a 800x600 display at 75Hz, meaning 75 frames are sent to the display every second.

   (a) What is the pixel rate? (On average how many pixels are sent to the display per second?)

   (b) For 24-bit color, what is the data rate?

   (c) The display has blanking intervals where the beam must go across the screen to start the next line or frame. Each horizontal line is scanned left-to-right, from top line to bottom line. The horizontal blanking interval is like writing additional invisible pixels to the end of the line, and the vertical blanking interval is like writing additional invisible lines to the end of the frame.



It turns out that the horizontal blanking interval, which occurs after each line, is 5.17171717μs. The vertical blanking interval, which occurs after each frame, is 0.5333333ms. What is the pixel frequency (clock used by display) for sending pixels to our 800x600 @ 75Hz display?

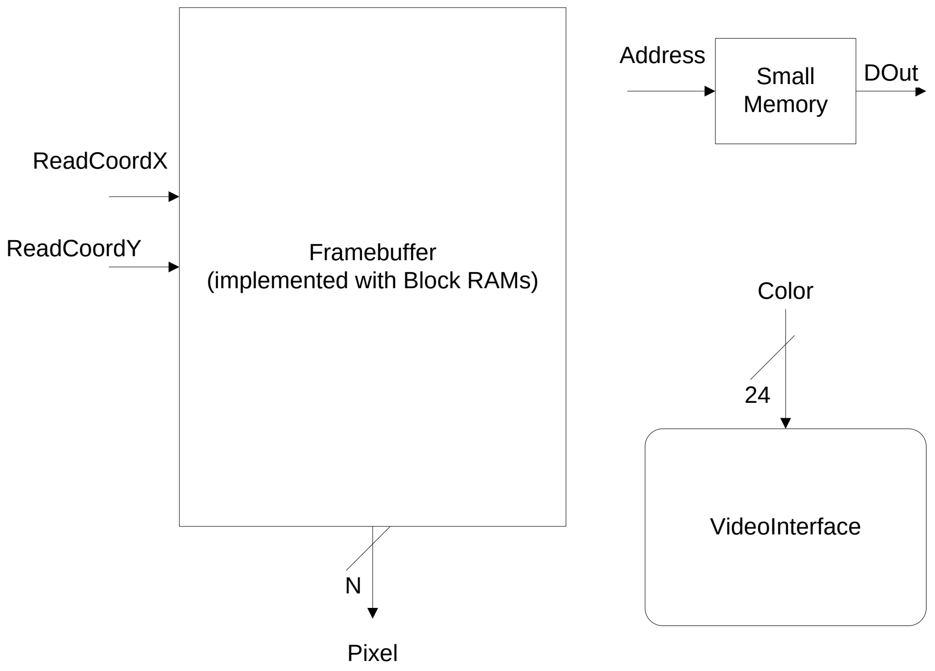
(d) What is the horizontal blanking interval in terms of number of pixels? What is the vertical blanking interval in terms of number of lines?

5. Suppose we implement our 800x600 pixel framebuffer using the Block RAM resources on our XC5VLX110T FPGA. You can look in "Virtex-5 Family Overview" on the Documents page of the website for the resources available on different Virtex-5 FPGA models. Suppose half the Block RAMs on the fpga are being used for our instruction and data memories already, leaving half for the framebuffer.

   (a) How many bits are available for each pixel?

   (b) How many distinct colors can be represented? (Let this number be hereafter called N)

   (c) Recall that our video interface supports 24-bit color. How can you use a relatively small memory to allow any N of the 2^{24} displayable colors to be used at a given time? Draw a high level block diagram of the framebuffer, video interface, and the extra memory block implementing this functionality. Specify the dimensions of the small memory.



   (d) The CPU should be able to write to this small memory (sometimes called a "Colormap") to change the colors being used and even do short animations efficiently. Suppose the Colormap is in the CPU memory map at addresses 0xA0000000 - 0xA000007C (remember memory is byte addressed but we only use word loads and stores so the 2 least significant bits of the address are always zero). Suppose initially both the framebuffer and Colormap are filled with 0's in every memory location. Fill in the code in main() that does an animation of a green vertical bar moving across the screen, with a black background. Make the animation occur at roughly 25 frames per second. Notice helpful functions sleep() and frameBufferAddress() are provided.

```
void sleep(long delay) {
    // ...makes CPU wait for 'delay' milliseconds...
}

int frameBufferAddress(int x, int y) {
    // ...returns the framebuffer memory address
    // corresponding to pixel (x,y)...
}

int main() {
    int BLACK = 0x000000;
    int GREEN = 0x00FF00;
    int colorMapBaseAddress = 0xA0000000;

    // write bars to the framebuffer, with color values 1 thru N-1.
    for (int x=1; x<N; x++) {
        for (int y=0; y<600; y++) {
            *(frameBufferAddress(x, y)) = x;  // write x to pixel (x,y)
        }
    }

    // Animation - animate a GREEN vertical bar moving across the screen
    // from x=1 to x=N-1.
    /* YOUR CODE HERE */
}
```