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

EECS150, Spring 2013

Homework Assignment 6: Video & Accelerators

Due March 12^{th}, 2pm

1. For the Xilinx device we use in the lab (LX110T), what is the total amount of on-chip SRAM bits available to the user (not counting the configuration bits) ? You can look in the Virtex-5 Family Overview on the Resources page of the website for the resources available on different Virtex-5 FPGA models. Now counting BlockRAM only, what is the total memory read bandwidth, in bits/second assuming assuming the maximum BlockRAM cycle rate is 400MHz.

2. We are trying to run an image processing application on our FPGA. According to the pseudocode (given below), the application updates every pixel according to the pixel to its left as well as the two pixels directly above and below it. it would need to do this update twice (after updating all the pixels, the code goes through every pixel to do the update again to produce the final result). You cannot make any assumption about the nature of the function f in the pseudocode. Also, column 0 is the left most column, pixel 0 is the top most pixel in a column.

```
For 2 iterations
   For column 0 to 1023
      For pixel 0 to 767
         A = value of the pixel above;
         B = value of the pixel below;
         C = value of the pixel to the left;
         pixel Value = f(A,B,C);
      endFor
    endFor
endFor
```

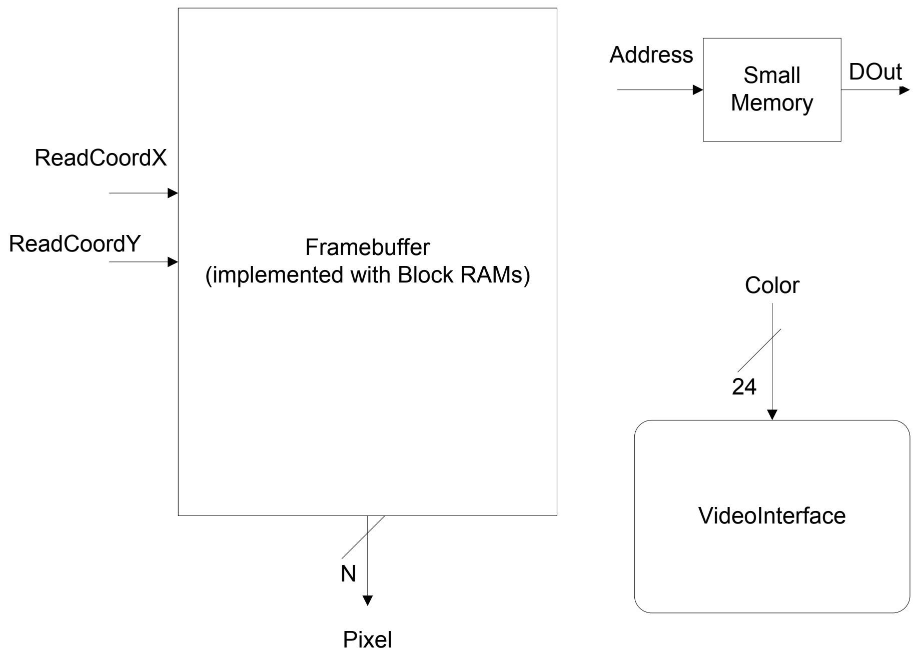
   The image is stored in DRAM. Each pixel takes up one 32-bit word, and there are 1024x768 pixels in a frame. Assume you have half of the BRAM available on the FPGA, and you can transfer eight 32-bit words (of consecutive DRAM addresses) between DRAM and BRAM every 200 ns, devise a strategy to minimize the time spent on data transfer. Using your strategy, how much time would be spent in transferring pixel data in and out of the on chip BRAM, for processing one frame (assume the BRAM is initially empty and the final result needs to be written to the DRAM)? Remember to specify if you are storing the pixels in DRAM in column major order or row major order, when do you transfer the data, and which part of the image is being transferred (which row/column).

3. Suppose we implement our 1024x768 pixel framebuffer using the Block RAM resources on our XC5VLX110T FPGA. Suppose a quarter of the Block RAMs on the fpga are being used for our instruction and data memories already, leaving three quarters for the framebuffer

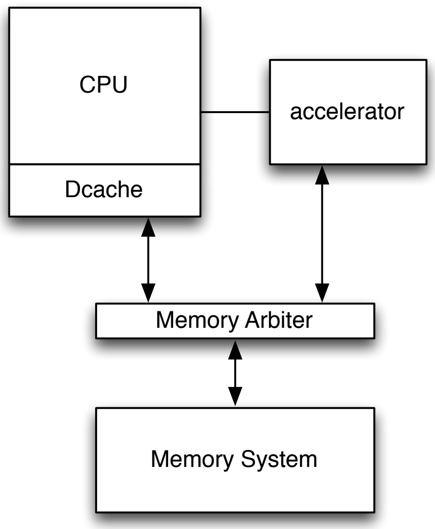
   (a) How many bits are available for each pixel?

(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.



4. We have a CPU+accelerator system, as given in the diagram below. The CPU runs at 100MHz while the accelerator runs at 200MHz. When the CPU or accelerator requests for access to the memory subsystem, it will not be granted the access until three CPU cycles after the other party finishes accessing the memory subsystem. The link between the CPU and the accelerator is a 32 bit wide asynchronous FIFO. Also, when the CPU or the accelerator is accessing the memory subsystem, it takes 100ns to transfer eight 32-bit words to/from the memory.



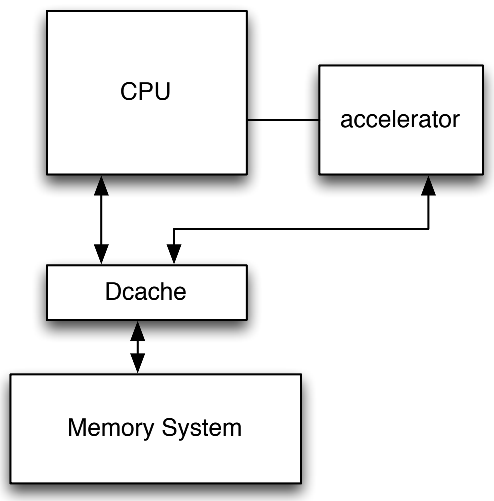
Now, we are going to process an array of 1024 words, which are currently in the Dcache(4KB cache) but not yet in the memory. The algorithm will look at each word in the array, perform a few arithmetic operations on it, and add the outcome to the sum of results from previous words in the array.

(a) One way to perform the computation will be using the CPU. Assume it takes one cycle to fetch every word, 7 cycles to perform the arithmetic operations, and one cycle to write a word. How long would it take to finish processing the array. (Hint: you can use register to store intermediate values, but we want the final result to be written to the cache)

(b) A second way to perform the algorithm is to have the CPU fetch a word, send it to the accelerator using the FIFO, and have the accelerator perform all the computations. When the final sum is produced, it will be sent back to the CPU and then gets written to the cache. It takes the CPU one cycle to fetch a word, one cycle to write it to the FIFO, and one cycle to write to the cache. It takes the accelerator one cycle to perform the arithemetic operations on up to 16 elements and add the results to the sum. Also it takes the accelerator one cycle to write the result to the FIFO. How long would the system take to finish processing the array? (You can ignore the delay introduced by the asynchronous FIFO itself)

(c) A third way to perform the algorithm is to have the array first transferred to the memory from the Dcache, and the accelerator can get the data from the memory. After the accelerator is done processing the data, it writes to the 32-bit wide FIFO, giving the final result back to the CPU, who would then write it to the Dcache. How long would this scheme take? You can use the performance parameters in the last part of the question.

5. We have a second CPU+accelerator system as shown below. The CPU is running at 50MHz and the accelerator is running at 100MHz as the shared cache negatively impacts their critical path. We are given the same algorithm as in the previous question, and the numbers of cycles the modules take to perform the operations are also the same as the last question. Again assume the array is in the Dcache but not in the memory.



(a) How long would it take for the CPU to process the array?

(b) Assume the accelerator can fetch/write one word at a time from/to the Dcache, and it takes one cycle for it to do so, how long would it take for the accelerator to process the array?

(c) If we use the CPU to fetch data and write to the cache, and use the FIFO to communicate the data and result between the CPU and the accelerator, how much time would it take to process the array?(Again ignore the delay introduced by the FIFO itself)