EECS150 Fall 2002 Checkpoint 4 (Draft)
UNIVERSITY OF CALIFORNIA AT BERKELEY
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

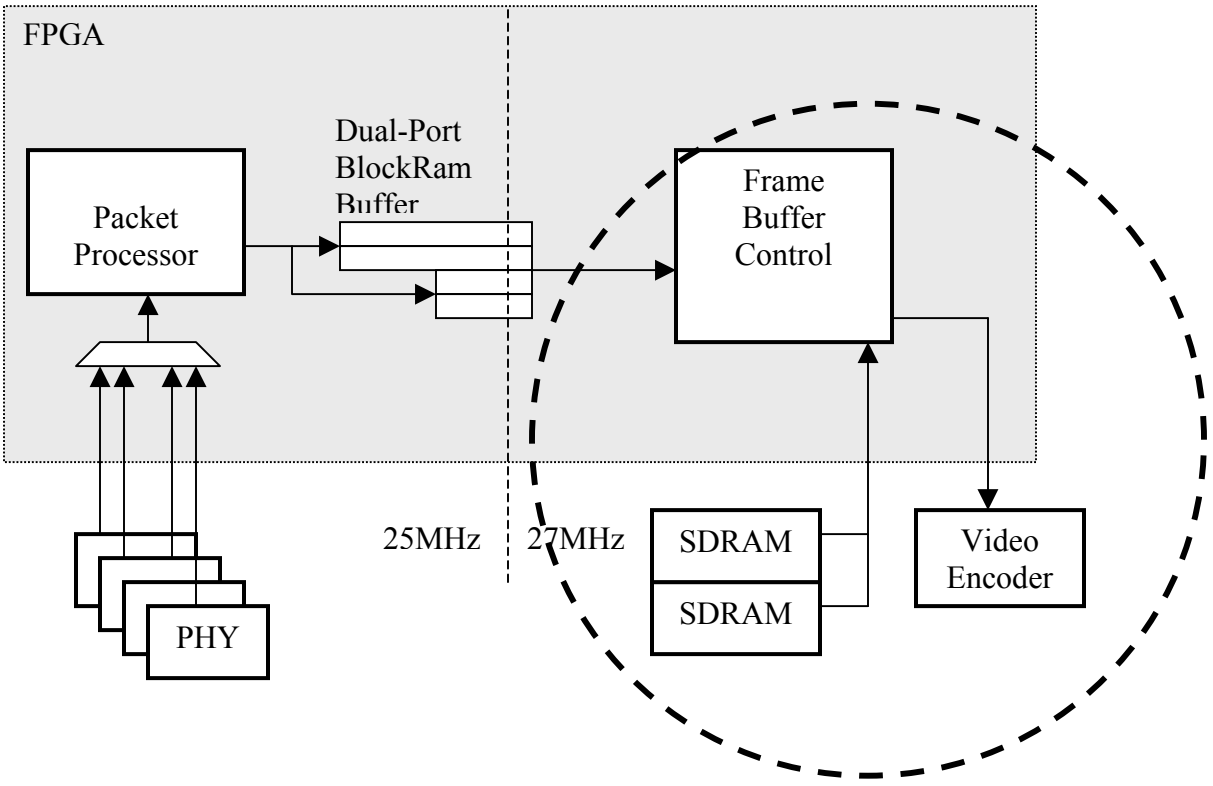
Checkpoint 4 **(DRAFT)**
Frame Buffer & Video Turtle

1 Introduction

This is the first checkpoint where you are expected to design a fairly large portion of the project (~ 1/3 to 1/2). As a large portion of this checkpoint can be reused in the project, coming up with a clean and elegant design and interface is very important. (Take the time to sit down and think about a good approach) **Please note that completing this checkpoint is a highly non-trivial task.** If the staff had to invest fair quantities of time into arriving at a working solution, then you can expect to do the same.

2 Overview

For this checkpoint we are going to be focusing on the frame buffer side of the project. You will need to make use of your knowledge of SDRAM and the video encoder.



You will use the SDRAM to store, at minimum, all the YUV values of the active screen. This is commonly referred to as a frame buffer, as it is a buffer that stores all the information about a frame in the video stream. You will read from the SDRAM frame buffer to get data to the video encoder, and any change you wish to make to the video should be written in to the frame buffer.

The data being written into said frame buffer for this checkpoint is the information associated with a cursor, or 'video turtle', the color, position, and movement rate of which are controlled via the buttons and dip switches. The movement of this turtle produces a trail on the display, yielding a result not entirely unlike an electronic etch-a-sketch.

3 Specifications

A solution to this checkpoint must do the following:

A) Take for input an Xmin, Xmax, Ymin, Ymax , Y, Cr, and Cb. The Xmin, Xmax, Ymin, Ymax, specify a rectangular subsection of the viewable screen area (as addresses in the frame buffer) in which the luminance and color values Y, Cr, and Cb, are to be written. The rest of the screen should not be written to and therefore will maintain its old value. If data is written out at any time other than that corresponding to the rectangular subsection of the screen specified by the Xmin, Xmax, Ymin, and Ymax, of the cursor, then previous data will be overwritten. **NOTE:** as the values for the corners of the cursor are given in terms of memory in the frame buffer, you must take some care to remember that the video display is interlaced, whereas the SDRAM address space is not, and compute membership in the rectangular region in a way that is conscious of whether one is processing the even or the odd field at the time.

B) Modify the color, location, and rate of movement of the turtle via button inputs that modify Xmin, Xmax, Ymin, and Ymax and the two banks of dip switches that determine color and rate of movement. The least significant four bits of the pair of dip switch banks control luminance, the next four, the value of Cr, the subsequent four, the value of Cb, and the remaining four bits control speed of motion. When cursor speed is set to values greater than one, it will move the cursor multiple units in the desired direction, **not** filling in the line segment in between.

C) Support a blanking of the video display. Because DRAM bits initialize to an unknown value, you will need to clear the frame buffer to all black YCrYCb = (32'h 10 80 10 80) before you proceed. Otherwise, the random values in the DRAM could be masquerading as EAV or SAVs in the middle of your active line.

To assist you in realizing these goals, the common modules file has been extended. Please read through this file, as we believe it to be of use to you.

4 Design Suggestions

Design suggestions: (this section is just some useful hints about the design, you do not have to follow this advice)

- Like checkpoint 2, you will need to use counters to keep track of pixels within a line and lines within a frame, plus additional signals to tell you when to send EAV, hBlank, SAV, active line, even field odd field…

- As there are many regions in the frame, you may find it useful to use an FSM type structure to keep track of where you are on the screen. Another approach would be to have a single bit for every region in the frame and used the pixel counter and line counter to set and reset the appropriate bits at the right cycle. If you chose to do this you might want to look at a SET/RESET flip-flop (an implementation of which is given in the common mods file) or a J/K flip-flop.

- The simplest way of dividing up the clock cycles between read and write is to just statically give every other 8 cycles to read or write. You have to be careful, however, because the number of cycles in a line is not a multiple of 16 cycles.

- Because the SDRAM has a 4 cycle latency between the active command and valid data output, you must start your read cycle before you need the data. Also it is always wise the register the data from the SDRAM at least once before sending them to the video encoder, if you do so, it will add an additional 1 cycle of latency.

- Although the SDRAM averages 1 byte per cycle it bursts 32 bits of data per cycle in 4 cycles and outputs nothing for the next 12 cycles. The video encoder, however, needs a continuous 8bits of data per cycle. Therefore, you will need some kind of buffering between the SDRAM and the video encoder.

- In order to satisfy setup and hold time constrains, the clock sent to the SDRAM is usually inverted relative to the operating clock for the express purpose of having changes not occur on the rising edge of the clock.

- There are many things that your design needs to do, so make sure you understand exactly what those things are before you start designing. Remember that there is no way you are going to design your circuit correctly if you don't know exactly what it is supposed to do. **Try drawing some timing diagrams of what exactly is supposed to happen and when.**

5 Acknowledgments

J. Wawryznek, N. Zhou, J. Sampson.
UCB 11 - 2002
EECS150 Fall 2002 Checkpoint 4

6 Checkoff

Name: _____ Name:_____

Section: _____

DATE_____TA:_____/_____(100%)