UNIVERSITY OF CALIFORNIA AT BERKELEY
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Checkpoint 2
Video Encoder

1.0 Motivation

This checkpoint serves three purposes:

1. Create a video encoder module for use in your painting game
2. Acquaint you with digital video in general and NTSC (TV) in particular
3. Provide your first major FSM design problem

With the first goal in mind you should make sure to **design your Verilog ahead of time**, comment you code and **test everything thoroughly**. Because you will be keeping and **relying on this code for months**, it will actually **save you many stressful hours to ensure it works well now**, rather than when you are about to finish the project.

Many of you will be at least reasonably familiar with video as it is used on computer systems, with standard VGA or DVI interfaces, complicated video cards and high-resolution data grade CRT or LCD monitors. In this checkpoint you will be introduced to the far older and more universal world of broadcast video. You will be working with NTSC standard video, as used in the U.S. broadcast TV industry, and in fact on your TV at home.

NTSC is a reasonably simple standard, which is widely used and readily available. But one of the primary reasons to use it is that despite advances such as DVI and HDTV, the video standards we will be using (which have survived from the invention of black-and-white television) will continue to affect video for a long time to come.

"BECAUSE YOU WILL BE KEEPING AND RELYING ON THIS CODE FOR MONTHS, IT WILL ACTUALLY SAVE YOU MANY STRESSFUL HOURS TO ENSURE IT WORKS WELL NOW, RATHER THAN WHEN YOU ARE ABOUT TO FINISH THE PROJECT"

2.0 Introduction

The module you will building for this checkpoint is a simple protocol bridge, connecting a memory structure, in this case a simple ROM-like module, to a byte wide video data stream.

The primary responsibilities of your module are:

1. Request the video data to be transmitted
    a. Must track which line and pixel is being sent when
2. Generate video framing signals
    a. Start-of-active-video and end-of-active-video (SAV and EAV) signals to delineate row boundaries
    b. Blanking and field control to indicate what type of data will be sent

   c. Black data to fill blanking periods
3. Clip the video data to the allowable value range
   a. Data values less than 0x10 or more than 0xF0 must be clipped
4. Initialize and control the Analog Devices ADV7194 hardware
   a. The ADV7194 is a digital-to-analog converter which generates analog video signals
   b. Use I^2C to initialize the ADV7194 to the proper mode

Your module will be responsible for abstracting away all the complexities of dealing with the ADV7194 and properly framing the data, leaving only the task of supplying valid video data, which for this checkpoint will be provided by a simple ROM-like module to generate a test pattern of solid color bars.

2.1 ITU-R BT.601 Video

When television broadcasts were slated to move from black-and-white to color, there was a significant concern over the existing investments in black-and-white sets. Not only were engineers concerned that old TV sets would be unable to receive newer signals, making old equipment useless, but they were worried that signals from older stations would not be received by the new color sets, preventing the adoption and sales of color TVs. As such a compromise was made resulting in the color scheme outlined below.

2.1.1 RGB Coloring & Human Perception

The standard color scheme used when dealing with light, as we are in video applications, is based on the three primary colors: Red, Green and Blue.

Human eyes have two sets of photoreceptors which detect incoming light:

- Rods cannot detect color, but they are very sensitive to brightness
- Cones can detect color, but they are significantly less sensitive

The primary colors Red, Green and Blue derive from the fact that cones come in three colors: Red, Green and Blue. This means that rather than generating any possible color, it is enough to be able to mix varying proportions of Red, Green and Blue, as our eyes perceive every other color in terms of the RGB proportions in them.

Just as important is the relative sensitivity of the rods and cones in our eyes, for example, because cones are not particularly sensitive, it is more than sufficient to store 8bits of intensity for each color in RGB, leading to the widespread use of 24bit color.

Less known but even more important is the fact that the rods in our eyes, which are sensitive only to brightness, are much more sensitive. This means that while we can easily perceive slight differences in brightness using our cones, it is more difficult to perceive subtle shades of color.

2.1.2 YUV Coloring

As a result of the economic pressures to maintain compatibility between older black and white TVs and the newer color models, as well as the way in which humans

perceive light, engineers designed a video format which would transmit intensity, or luminance, and color, or chrominance, separately.

This means that instead of storing the Red, Green and Blue content of a pixel in our video, we will store its luminance (Y) or brightness and its red (C_R/V) and blue (C_B/U) chrominance, or color.

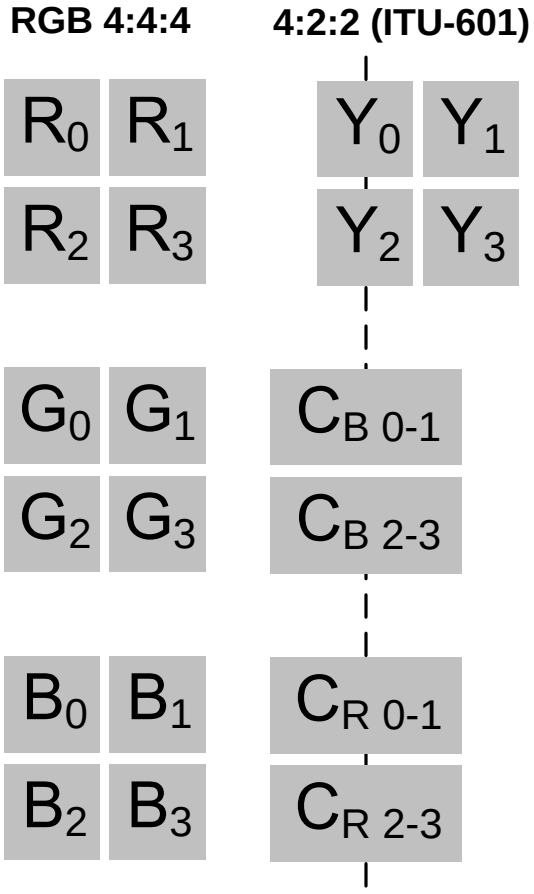

Figure 1: RGB/YC_RC_B with Sub-Sampling

As shown above, we also can take advantage of the fact that the cones in the human eye are significantly less sensitive, by storing only half as much color information as brightness information, as shown by the chrominance sub-sampling map in Figure 1.

In order to transmit the data serially, it is reduced to pixel pairs, each of them 32bits wide with two luminance (Y) values, and one each of the chrominance values, red (C_R) and blue (C_B).

Line i-1: C_B Y C_R Y C_B Y C_R Y C_B Y C_R Y ...
Line i: C_B Y C_R Y C_B Y C_R Y C_B Y C_R Y ...
Line i+1: C_B Y C_R Y C_B Y C_R Y C_B Y C_R Y ...

2.2 ITU-R BT.656 Video

The ITU-R BT.601 standard outlined above covers how the pixels are sampled digitally and encoded with brightness and color information. The ITU-R BT.656 standard outlines how to organize, frame and transmit the data.

Because these video standards are meant to be the digital equivalent of the older analog standards, they retain the framing and format of the analog signals. Because of this, the job of the ADV7194 video encoder on the CaLinx2 boards, is entirely to perform a digital to analog conversion. As a result, it is the job of your VideoEncoder.v module to add all the framing signals.

ITU-R BT.656 is meant to convey interlaced video signals, along with the blanking periods which old CRT TV sets used to align and move their electron guns.

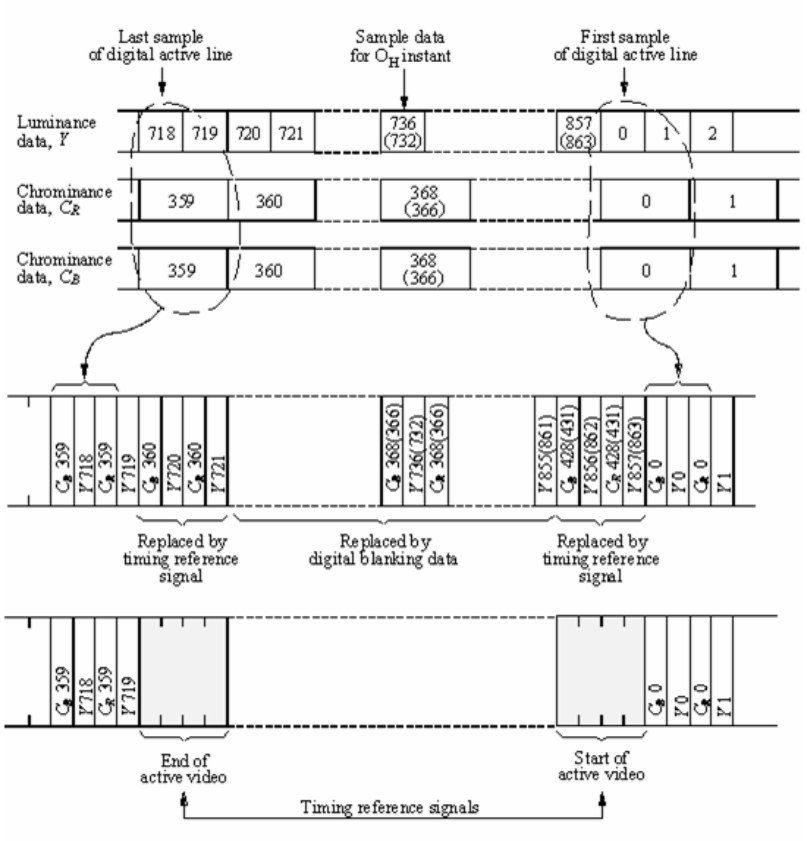

Figure 2: ITU-R BT.656 Format Detail

Shown in Figure 2 above is a detailed view of horizontal blanking interval between two lines of active video data. Shown in Figure 3 below is an overview of the format of an entire frame, including both odd and even fields, and horizontal and vertical blanking intervals.

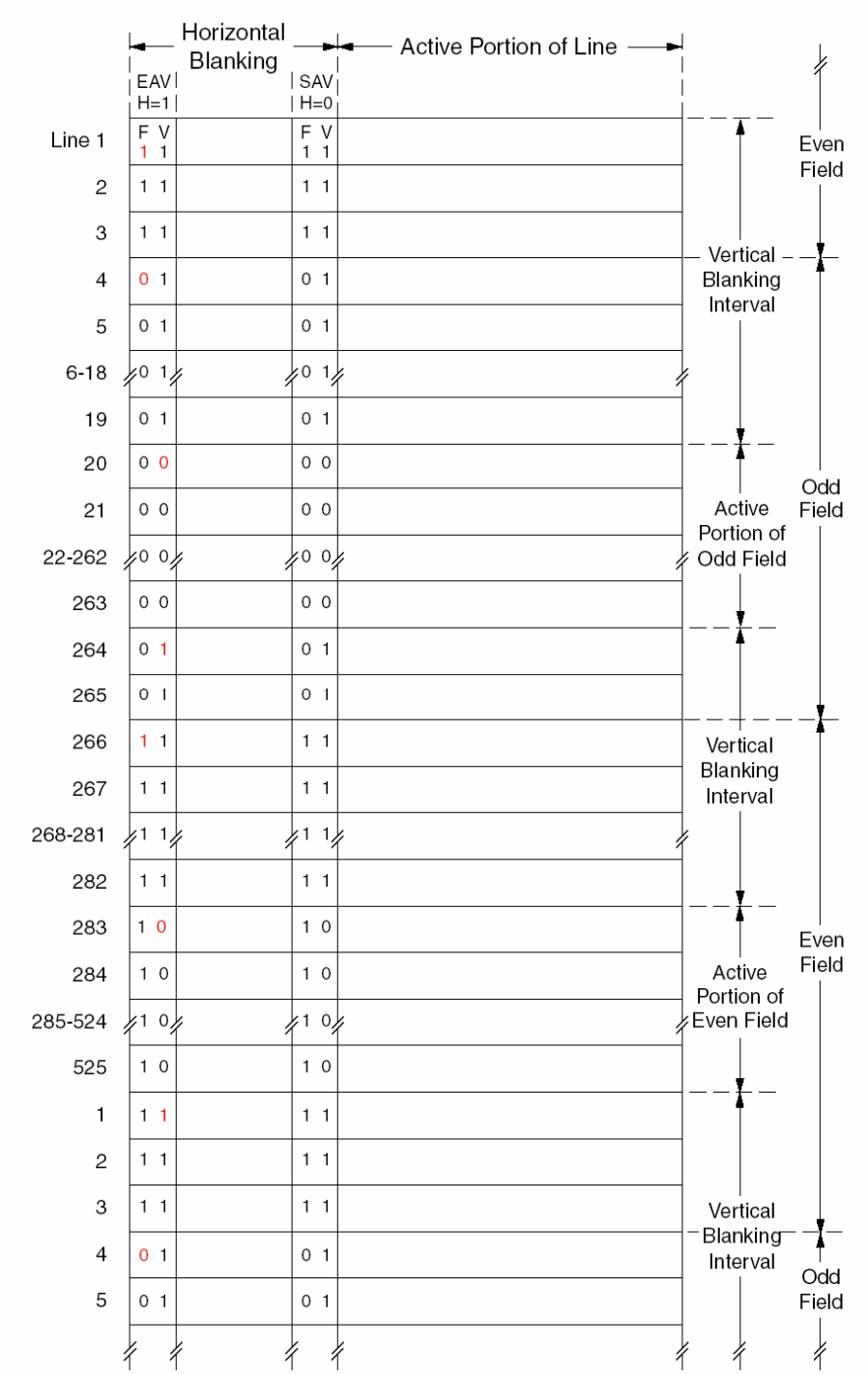

Figure 3: ITU-R BT.656 Video Frame Overview
(Notice this figure only shows 487 active lines!)

ITU-R BT.656 Video is essentially a stream of lines of video, as shown in Figure 3 above. Each line begins with an EAV (counter intuitively), followed by a horizontal blanking interval, followed by an SAV and then, if the line is not in a vertical blanking interval 720 pixels (1440 bytes) of active video data.

The EAV and SAV codes are detailed in the Video in a Nutshell document on the EECS150 website. Please refer to that or the lab lecture for more details.

3.0 Prelab

Please make sure to complete the prelab before you attend your lab section. **You will not be able to finish this checkpoint in 3hrs! Labs are getting progressively longer, groups averaged roughly 10hrs for Lab5.**

1. **Read this handout thoroughly**.
    a. Pay particular attention to section **4.0 Lab Procedure** as it describes what you will be doing in detail.
2. Examine the documents page of the website
    a. **You will need to get used to reading datasheets, like these. They form the core of information available to hardware designers.**
    b. http://www-inst.eecs.berkeley.edu/~cs150/fa04/Documents.htm#Video
    c. **Read the ADV7194 Datasheet**
    d. Read Video in a Nutshell by Tom Oberheim
    e. Read the ITU-R BT.656 Standard
    f. Read the ITU-R BT.601 Standard
3. **Examine the Verilog** provided for this checkpoint
    a. There isn't much, so it should be pretty clear
    b. You do not need to understand the I^2C module in full detail
4. **Start your design ahead of time**.
    a. Begin with **schematics** and **bubble-and-arc** diagrams
    b. Come prepared for your design review
    c. **VideoEncoder.v** will require significant debugging to finish. Make sure to write at least a draft of it ahead of time.
    d. Start building your testbench early
        i. Perhaps have one person design the module and the other design the testbench
5. You will need **at least the entire 3hr lab**!
    a. You will need to test and debug your verilog thoroughly.
    b. **You must build a reliable interface with a real hardware component!**

4.0 Lab Procedure

Remember to **manage your Verilog, projects and folders well**. Doing a poor job of managing your files can cost you **hours of rewriting code**, if you accidentally delete your files.

4.1 VideoEncoder.v

This is the **main module you will need to build** for this checkpoint. Shown in figure 4 below is one possible block diagram, you may start your design from.

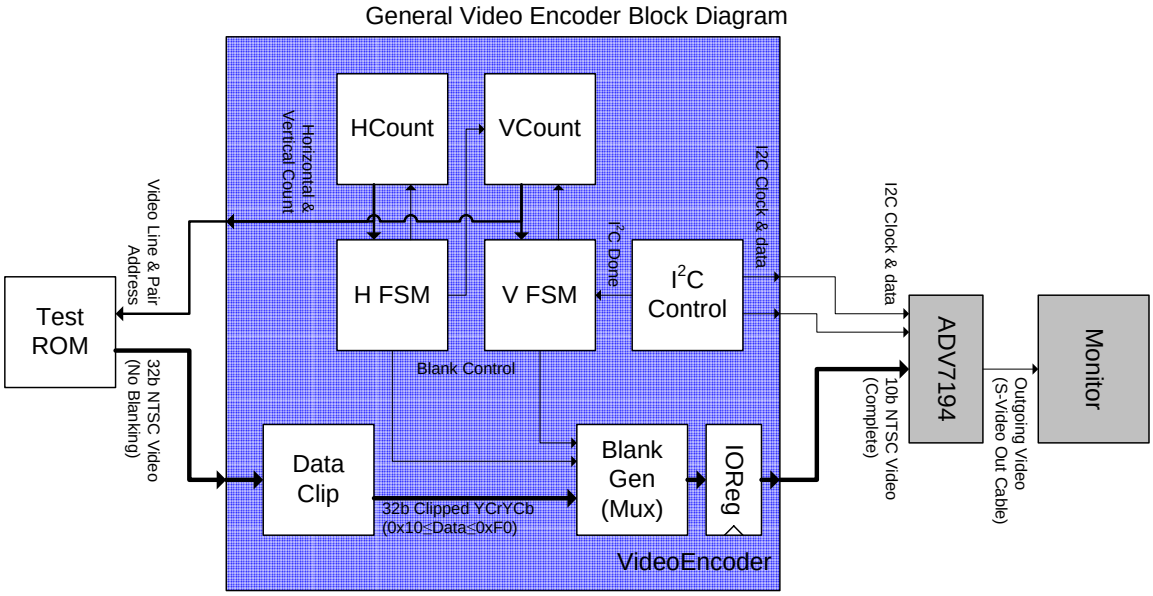

Figure 4: Starting Block Diagram for VideoEncoder.v

The essential functions of you module are listed in section 2.0 Introduction, above. Note that out of this list, we have provided you with the I^2C interface, in an effort to keep the checkpoint manageable. The primary reason for this is that if the ADV7194 chip is improperly initialized, you will not see ANY video on the board, making debugging rather difficult.

Shown in Table 1 below is the port specification list for VideoEncoder.v.

Signal	**Width**	**Dir**	**Description**
VE_P	10	O	Outgoing NTSC Video (Use {Data, 2'b00})
VE_SCLK	1	O	I^2C Clock (For Initialization)
VE_SDA	1	O	I^2C Data (For Initialization)
VE_PAL_NTSC	1	O	PAL/NTSC Mode Select (Always 1'b0)
VE_RESET_B_	1	O	Active low reset out the ADV7194
VE_HSYNC_B_	1	O	Manual sync control (Always 1'b1)
VE_VSYNC_B_	1	O	Manual sync control (Always 1'b1)
VE_BLANK_B_	1	O	Manual sync control (Always 1'b1)
VE_SCRESET	1	O	Manual sync control (Always 1'b0)
VE_CLKIN	1	O	Clock (27MHz, Clock Signal)
Clock	1	I	The Clock signal
Reset	1	I	Reset all counters and registers (Not necessarily to 0)
Din	32	I	Requested video data from ROM, YC_RYC_B
InRequest	1	O	Request signal, should be high for one out four cycles during active video periods. DIn will be valid after the riding edge where this is 1'b1.

InRequestLine	9	O	The line from which you are requesting video data, (Use {Line, Field})
InRequestPair	9	O	The pair of pixels which you are requesting from the video ROM. Notice that this address should go up by one for each 4-byte pixel pair.

Table 1: Port Specification for VideoEncoder.v

- You may fill the blanking interval with any video data you choose, but black is best: Luma 0x10, Chroma 0x80
- It is not necessary to start transmitting an EAV immediately when the I^2C interface is done, as indicated by the I2CDone signal.
    - The ADV7194 will wait until it gets an EAV (0xFF, 0x00, 0x00, code) before it begins decoding your data.
- Once you do begin sending, there is a very tight constraint on what you send on which cycle, **check your counters carefully** as even a small mistake may cause major problems.
    - A shaky or scrolling picture most likely means one of your many counters is off by a few cycles. Check your state machine/counter logic.
    - You may wish to design your testbench to count cycles too, in an effort to automatically double check your work.
- Weird colors most likely mean that you are transmitting the Y, CR and CB values in the wrong order.
    - Lots of green and purple means you have swapped the luma and chroma values.
    - Wrong values of Y,Cb and Cr could also result in the monitor losing sync, make sure to clip all actual video data to the range 0x10 – 0xF0.

5.0 Checkpoint2 Checkoff

Name: _____ Name: _____
Section: _____

I Simulation of Video Encoder _____ (40%)
II Display Test Pattern on Board _____ (60%)

III Total: _____
IV TA: _____

V Hours Spent: _____

RevB – 10/14/2005	Brian Gawalt	Built from Checkpoint2 from Fall2004
RevA – 10/14/2004	Greg Gibeling	Built from Checkpoint2 from Fall2003 Pieces extracted from Video in a Nutshell by Tom Oberheim Pieces extracted from EECS150 lecture slides by John Wawrzynek