EECS150 Final Project, Fall 1999: Digital Image Manipulator

R.S. Fearing and G. Moy
Dept. of Electrical Engineering and Computer Sciences
Univ. of California, Berkeley

1 Objectives

The final project for this semester is to design a digital image manipulator (DIM). The DIM will receive data from a video camera or serial port, manipulate the image, and display the new image on an LED array. In the first mode of operation, the data will be received over a 115.2 KBaud serial line, stored in a 64K byte RAM, and displayed on the LED array (10 rows, 14 columns). In the second mode, data from either the camera or serial port is stored in the RAM, manipulated, then displayed on the LED array. Manipulations include panning, zooming, and flipping. In the third mode, multiple frames from the camera or serial port are stored in the RAM and played back as a movie. The DIM could be used as an image magnification device for the severely visually impaired, or to obtain high speed visual effects not obtainable with conventional LCD or CRT displays.

1.1 General Philosophy

This document describes the input/output specification for the DIM, and gives the schedule for completing requirements for this project. As in the real world, the user/external interface is specified; it is up to you to specify most of what goes into the system. You can consider the course staff as being the customers for your project, and you have contracted to deliver a working system. We have checkpoints and demonstrations along the way to see that our contract will be satisfied. Our contract also has a clause that you won't get paid (i.e. credit) if you don't deliver the working pieces on the specified dates. Exceptions can only be made for serious medical problems. Now you may want to deliver extra features, which is fine, but the basic system needs to be working first. As customers, we aren't going to pay for simulations, we need to see the real thing.

1.2 General Tips

Just because a design works does not make it a good design. Use good design practices such as:

- top-down design
- design, simulate, and debug in modules
- use synchronous design methodology (e.g. use only synchronous reset parts). (One global asynchronous reset for the whole design using the STARTUP block is fine). Use BUFGS for a single global clock. Pay attention to timing for portions of the system running at more than 5 MHz, (use Xilinx timing analyzer tool to check for worst case path).
- divide work up with your partner for individual modules (but make sure you have agreed on the interface!)

Budget plenty of time to get your project done. With this level of complexity, it takes at least three times as long to debug a system as it does to design and enter it. In fact, however long you think it will take to complete a task, it will take 3 times longer. Plan accordingly. Also, lab space is limited, and workstations will need to be shared. Plan accordingly.

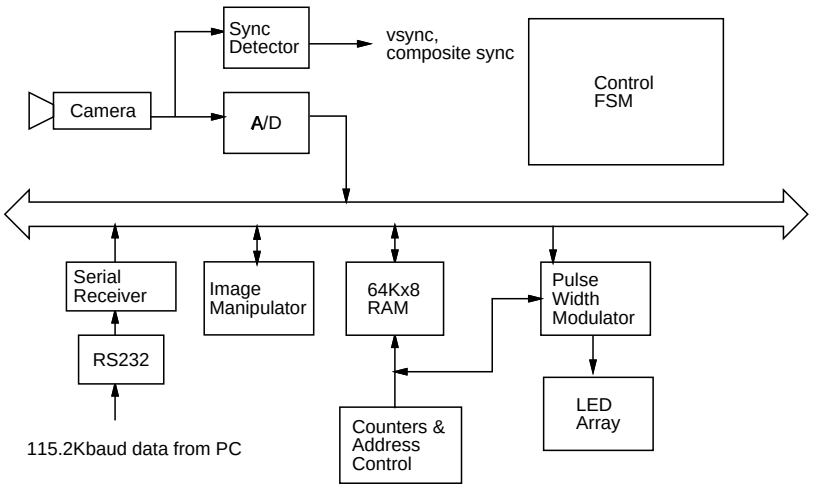


Figure 1: Top Level Block Diagram.

2 Project Description

A top level block diagram of the project is shown in Figure 1. (You are free to rearrange the blocks to optimize or simplify the design. For example, the video A/D never drives the PWM directly.) There are six main blocks in the system:

1. Memory and address block. A 64K byte RAM will be used to store video data, an image of size up to 256 by 256. You can think of the memory as holding an array:
   `unsigned char image[256][256];` .
   The counters are used to sequentially store and retrieve data from the memory. This module is built in Lab 7.

2. The serial receiver receives 8 data bits with a start and stop bit from the PC serial port at 115.2KBaud. The effective data rate is about 11 K Bytes per second. This block will be built and debugged for checkpoint 1.

3. The pulse width modulation (PWM) module converts a brightness (image data value) to a waveform of corresponding duty cycle used to drive the LED Array. The percentage on-time or duty cycle of the LED controls its brightness. This block will be built and debugged for checkpoint 2.

4. The video interface converts the analog data from a video camera into 8 bits of digital data (one byte per pixel) using an analog to digital (A/D) converter. The *sync detector* provides synchronization information, in particular, when the top of the image is (vertical sync) and when the start of a line is (horizontal sync). This block will be built and debugged in checkpoint 3.

5. The image manipulator will implement your image manipulation features on the image data stored in memory. Different combinations of features will be assigned. Part of the project is designing your own algorithm which fits in the XC4005.

6. The controller module is responsible for generating all clock, timing, user input/output, and control signals. It is probably better to have a number of simple, modular FSMs controlling various functions, rather than one huge FSM controlling everything.

2.1 General Features

The image data will come from either the camera or serial line. There will be two image sizes for this project. The full (large) frame is 210 rows and 196 columns. The small frame is 10 rows and 14 columns. Your control FSM should run at 16MHz, 18MHz, or 20MHz, as assigned. For the image manipulation, use even subsampling over the area of interest.

2.2 Operation Mode 1: LED display

In mode 1, the system operates as a gray-scale LED display. Data from a single 10 by 14 small frame is received from the serial interface, stored in the RAM, and displayed on the LED Array. This mode can be used to verify that the serial transmitter, RAM, PWM module, and LED array are functioning correctly.

2.3 Operation Mode 2: Image Manipulator

In mode 2, the system operates as a digital image manipulator. Groups will implement their assigned features. For groups with serial input for the manipulation part, a large frame (210 by 196) will be sent from the computer over the serial line. For groups with camera input, use the camera to take a snapshot (210 by 196).

2.4 Operation Mode 3: Playback

Using a series of small frames received from the camera or serial line and stored in RAM, play the series of frames back as a movie on the LED array. Store at least 256 small (10 by 14) frames in memory. Repeat the playback until RESET or MODE changes. Each frame should be held for 1/60 second before displaying the next frame in the sequence. You may want to add single step, slow motion or reverse capability.

2.5 Clock Speed and Features

To increase the design space of projects, your team's clock speed and features will depend on student ID numbers. Add the last 3 digits together, resulting in a number from 0 to 1998. (Example 11112222 + 22223333 gives 555). Take the result MOD 126. (Example: 555 MOD 126 = 51). Re-entry students, please use 000. The assigned features and input sources are shown in the following table. There is not much difference in complexity between the different features. The purpose of this design variation is to make it a little more difficult for your project design to get ripped off.

Clock Speed and Feature Table

SID Sum	Clock Speed	Manip. Feature	Direction	Manip. Input	Playback Input	SID Sum	Clock Speed	Manip. Feature	Direction	Manip. Input	Playback Input
000	16MHz	Zoom	Upper Left	Serial	Camera	001	16MHz	Zoom	Upper Middle	Serial	Camera
002	16MHz	Zoom	Upper Right	Serial	Camera	003	16MHz	Zoom	Middle Left	Serial	Camera
004	16MHz	Zoom	Middle	Serial	Camera	005	16MHz	Zoom	Middle Right	Serial	Camera
006	16MHz	Zoom	Lower Left	Serial	Camera	007	16MHz	Zoom	Lower Middle	Serial	Camera
008	16MHz	Zoom	Lower Right	Serial	Camera	009	16MHz	Zoom	Upper Left	Camera	Serial
010	16MHz	Zoom	Upper Middle	Camera	Serial	011	16MHz	Zoom	Upper Right	Camera	Serial
012	16MHz	Zoom	Middle Left	Camera	Serial	013	16MHz	Zoom	Middle	Camera	Serial
014	16MHz	Zoom	Middle Right	Camera	Serial	015	16MHz	Zoom	Lower Left	Camera	Serial
016	16MHz	Zoom	Lower Middle	Camera	Serial	017	16MHz	Zoom	Lower Right	Camera	Serial
018	18MHz	Zoom	Upper Left	Serial	Camera	019	18MHz	Zoom	Upper Middle	Serial	Camera
020	18MHz	Zoom	Upper Right	Serial	Camera	021	18MHz	Zoom	Middle Left	Serial	Camera
022	18MHz	Zoom	Middle	Serial	Camera	023	18MHz	Zoom	Middle Right	Serial	Camera
024	18MHz	Zoom	Lower Left	Serial	Camera	025	18MHz	Zoom	Lower Middle	Serial	Camera
026	18MHz	Zoom	Lower Right	Serial	Camera	027	18MHz	Zoom	Upper Left	Camera	Serial
028	18MHz	Zoom	Upper Middle	Camera	Serial	029	18MHz	Zoom	Upper Right	Camera	Serial
030	18MHz	Zoom	Middle Left	Camera	Serial	031	18MHz	Zoom	Middle	Camera	Serial
032	18MHz	Zoom	Middle Right	Camera	Serial	033	18MHz	Zoom	Lower Left	Camera	Serial
034	18MHz	Zoom	Lower Middle	Camera	Serial	035	18MHz	Zoom	Lower Right	Camera	Serial
036	20MHz	Zoom	Upper Left	Serial	Camera	037	20MHz	Zoom	Upper Middle	Serial	Camera
038	20MHz	Zoom	Upper Right	Serial	Camera	039	20MHz	Zoom	Middle Left	Serial	Camera
040	20MHz	Zoom	Middle	Serial	Camera	041	20MHz	Zoom	Middle Right	Serial	Camera
042	20MHz	Zoom	Lower Left	Serial	Camera	043	20MHz	Zoom	Lower Middle	Serial	Camera
044	20MHz	Zoom	Lower Right	Serial	Camera	045	20MHz	Zoom	Upper Left	Camera	Serial
046	20MHz	Zoom	Upper Middle	Camera	Serial	047	20MHz	Zoom	Upper Right	Camera	Serial
048	20MHz	Zoom	Middle Left	Camera	Serial	049	20MHz	Zoom	Middle	Camera	Serial
050	20MHz	Zoom	Middle Right	Camera	Serial	051	20MHz	Zoom	Lower Left	Camera	Serial
052	20MHz	Zoom	Lower Middle	Camera	Serial	053	20MHz	Zoom	Lower Right	Camera	Serial

SID Sum	Clock Speed	Manip. Feature	Direction	Manip. Input	Playback Input	SID Sum	Clock Speed	Manip. Feature	Direction	Manip. Input	Playback Input
054	16MHz	Pan	UL → LR	Serial	Camera	055	16MHz	Pan	UM → LM	Serial	Camera
056	16MHz	Pan	UR → LL	Serial	Camera	057	16MHz	Pan	ML → MR	Serial	Camera
058	16MHz	Pan	MR → ML	Serial	Camera	059	16MHz	Pan	LL → UR	Serial	Camera
060	16MHz	Pan	LM → UM	Serial	Camera	061	16MHz	Pan	LR → UL	Serial	Camera
062	16MHz	Pan	UL → LR	Camera	Serial	063	16MHz	Pan	UM → LM	Camera	Serial
064	16MHz	Pan	UR → LR	Camera	Serial	065	16MHz	Pan	ML → MR	Camera	Serial
066	16MHz	Pan	MR → ML	Camera	Serial	067	16MHz	Pan	LL → UR	Camera	Serial
068	16MHz	Pan	LM → UM	Camera	Serial	069	16MHz	Pan	LR → UL	Camera	Serial
070	18MHz	Pan	UL → LR	Serial	Camera	071	18MHz	Pan	UM → LM	Serial	Camera
072	18MHz	Pan	UR → LL	Serial	Camera	073	18MHz	Pan	ML → MR	Serial	Camera
074	18MHz	Pan	MR → ML	Serial	Camera	075	18MHz	Pan	LL → UR	Serial	Camera
076	18MHz	Pan	LM → UM	Serial	Camera	077	18MHz	Pan	LR → UL	Serial	Camera
078	18MHz	Pan	UL → LR	Camera	Serial	079	18MHz	Pan	UM → LM	Camera	Serial
080	18MHz	Pan	UR → LR	Camera	Serial	081	18MHz	Pan	ML → MR	Camera	Serial
082	18MHz	Pan	MR → ML	Camera	Serial	083	18MHz	Pan	LL → UR	Camera	Serial
084	18MHz	Pan	LM → UM	Camera	Serial	085	18MHz	Pan	LR → UL	Camera	Serial
086	20MHz	Pan	UL → LR	Serial	Camera	087	20MHz	Pan	UM → LM	Serial	Camera
088	20MHz	Pan	UR → LL	Serial	Camera	089	20MHz	Pan	ML → MR	Serial	Camera
090	20MHz	Pan	MR → ML	Serial	Camera	091	20MHz	Pan	LL → UR	Serial	Camera
092	20MHz	Pan	LM → UM	Serial	Camera	093	20MHz	Pan	LR → UL	Serial	Camera
094	20MHz	Pan	UL → LR	Camera	Serial	095	20MHz	Pan	UM → LM	Camera	Serial
096	20MHz	Pan	UR → LR	Camera	Serial	097	20MHz	Pan	ML → MR	Camera	Serial
098	20MHz	Pan	MR → ML	Camera	Serial	099	20MHz	Pan	LL → UR	Camera	Serial
100	20MHz	Pan	LM → UM	Camera	Serial	101	20MHz	Pan	LR → UL	Camera	Serial

UL = Upper Left	UM = Upper Middle	UM = Upper Right
ML = Middle Left	MM = Middle Middle	MM = Middle Right
LL = Lower Left	LM = Lower Middle	LM = Lower Right

SID Sum	Clock Speed	Manip. Feature	Direction	Manip. Input	Playback Input	SID Sum	Clock Speed	Manip. Feature	Direction	Manip. Input	Playback Input
102	16MHz	Flip	Right to Left	Serial	Camera	103	16MHz	Flip	Left to Right	Serial	Camera
104	16MHz	Flip	Top to Bottom	Serial	Camera	105	16MHz	Flip	Bottom to Top	Serial	Camera
106	16MHz	Flip	Right to Left	Camera	Serial	107	16MHz	Flip	Left to Right	Camera	Serial
108	16MHz	Flip	Top to Bottom	Camera	Serial	109	16MHz	Flip	Bottom to Top	Camera	Serial
110	18MHz	Flip	Right to Left	Serial	Camera	111	18MHz	Flip	Left to Right	Serial	Camera
112	18MHz	Flip	Top to Bottom	Serial	Camera	113	18MHz	Flip	Bottom to Top	Serial	Camera
114	18MHz	Flip	Right to Left	Camera	Serial	115	18MHz	Flip	Left to Right	Camera	Serial
116	18MHz	Flip	Top to Bottom	Camera	Serial	117	18MHz	Flip	Bottom to Top	Camera	Serial
118	20MHz	Flip	Right to Left	Serial	Camera	119	20MHz	Flip	Left to Right	Serial	Camera
120	20MHz	Flip	Top to Bottom	Serial	Camera	121	20MHz	Flip	Bottom to Top	Serial	Camera
122	20MHz	Flip	Right to Left	Camera	Serial	123	20MHz	Flip	Left to Right	Camera	Serial
124	20MHz	Flip	Top to Bottom	Camera	Serial	125	20MHz	Flip	Bottom to Top	Camera	Serial

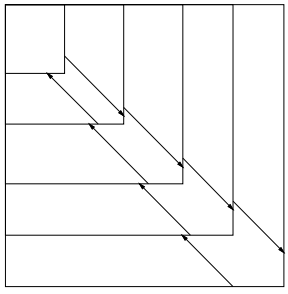
Mode 2 = Manip. Feature. Mode 3 = Playback Input

3 Feature Description

Each feature is subdivided into 4 steps. Since the LED Array updates very quickly (60Hz), you will need to refresh each step so that each step is visible for about 1 second (1/60 second for playback).

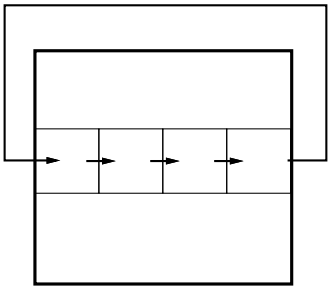
3.1 Zoom

Using a full frame, zoom into the assigned area in 4 steps. Zoom out of the area in 4 steps. Repeat until RESET or MODE changes. You will need to use evenly spaced pixels corresponding to the dimensions of the LED Array. Here is an example of the Zoom operation (Upper Left):



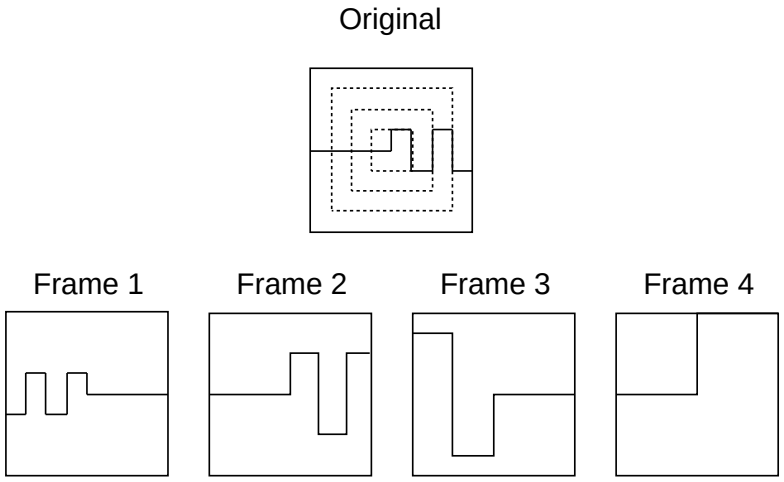
3.2 Pan

Using a full frame, pan across the image in the assigned direction in 4 steps. Repeat until RESET or MODE changes. You will need to use evenly spaced pixels corresponding to the dimensions of the LED Array. Here is an example of the Pan operation (Left to Right):



3.3 Flip

Using a full frame, flip the image in the assigned direction in 4 steps. The first step will use the whole image. The second frame will use the inner 3/4 of the image. The third frame will used the inner 1/2 of the frame. The last frame will use the inner 1/4 of the frame. Essentially, you are implementing a center zoom + flip). Repeat until RESET or MODE changes. Here is an example of the Flip operation (Left to Right):



4 Pulse Width Modulation

Pulse Width Modulation is a modulation scheme where the percentage of '1' to '0' output is proportional to the value evaluated. For example, for a 5-bit, 32 time slice PWM scheme, the following table can be used as the PWM converter:

```
                Time Slice
        00000000001111111111222222222233
        01234567890123456789012345678901
Value
  00    00000000000000000000000000000000
  01    10000000000000000000000000000000
  02    11000000000000000000000000000000
  03    11100000000000000000000000000000
  04    11110000000000000000000000000000
   .                    .
   .                    .
   .                    .
  31    11111111111111111111111111111111
```

The PWM module takes in a 5-bit number as an input. The output is either a '0' or '1', depending on the value and the current time slice being evaluated. The PWM module will read values from the RAM and output to the LED Array. Since the data uses 5 bits, 32 time slices are needed. The LED Array is controlled a row of 14 LEDs at a time. In other words, an LED brightness is specified by a number from 0 to 31. A brightness of 16 would mean that the LED is on approximately half the time, and would be perceived by the eye as half brightness. This module needs to do the following:

```
repeat until RESET or new MODE
for ROW = 0 to 9
 for TIME_SLICE = 0 to 31
  send 3 bits selecting ROW to LED Array
  for COL = 0 to 13
   VALUE = DATA(ROW,COL) from RAM
   decide on a 0 or 1 depending on VALUE and TIME_SLICE
   send bit to LED Array
  end
  set ENABLE to turn on row of LED Array
 end
end
```

4.1 LED Array Protocol

The checkpoint 2 handout will provide further details on the LED interface.

There are 4 interface pins on the LED Array circuit board. They are CLK, ENABLE, DATA, and CLEAR, in addition to power and ground.

CLK is used as the clock for the LED Array. It must be generated (glitch-free) from the Xilinx board and sent to the LED Array through a wire.

DATA is serially fed to an 18 bit shift register. The first 4 bits denote the row to be lit up. The next 14 bits specify whether a corresponding LED in the row will be turned on.

Example: 100011111111111111 means that during this time slice, row 8 will turn on all 14 LEDs.

ENABLE is used to enable the driver chips, turning on the LEDs after the serial data register has been fully loaded.

The LED array should be updated at the 60 Hz rate of TV signals. Thus an LED frame is updated every 16.7 ms and a line is updated every 1.67 ms. With 32 time slices per line, a time slice is 52.1 μsec. The CLK signal needs to be fast enough to send out all 18 bits in significantly less than 52.1 μsec, otherwise you won't get sufficient brightness on the LEDs, since ENABLE can only be asserted after all data has been loaded into the 18 bit shift register.

5 Serial Protocol

The data packets from the serial transmitter are in the form: **Header**, **Data**, ..., **Data**, **Footer**.

Figure 2 shows the data transmitted in the two modes. The header byte signals the start of a serial transmission packet by setting its MSB=1 (bit 7 = 1). (All data bytes in the packet have MSB=0). Bits 6 and 5 are 00 for a small frame and 10 for a large frame. A byte with the 3 MSB = 101 is a footer. A Data byte has the 3 MSB = 000, and the 5 LSB specify the brightness for that pixel.

For a large frame, there will be 41160 (210×196) bytes of data. For a small frame, there will be 140 (10×14) bytes of data.

See Checkpoint 1 for specifications of the serial receiver.

6 Video Interface

Basically, a television monitor displays a serial data stream in a parallel format. A television displays data by means of a *raster scan* . The electron beam which fluoresces the phosphor (in the cathode ray tube

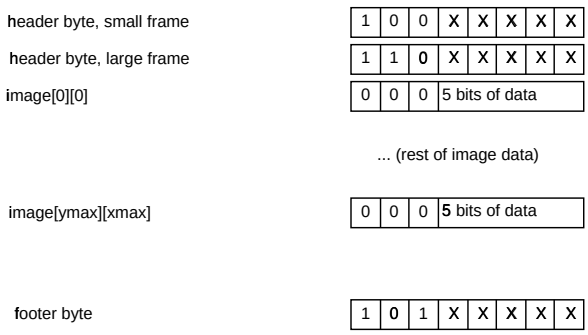


Figure 2: Serial transmission protocols for data reception. The MSB must be 1 in a header or footer byte.

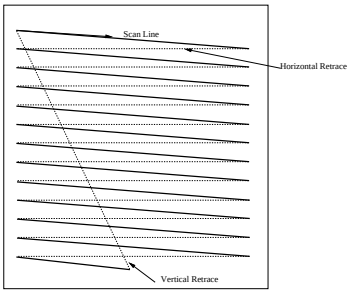


Figure 3: Raster Scan.

= CRT), sweeps across and down the picture tube in a zig-zag fashion, starting from the upper left-hand corner. (Fig. 3). (The electron beam is steered by electro-magnets mounted on the picture tube).

In the figure, the heavy lines are the display. Since the vertical sweep is continuous, these lines are not quite horizontal. The light lines are the horizontal and vertical *retrace* . Horizontal and vertical *blanking* are usually applied during the respective retraces to prevent the display of unwanted data. Unlike an oscilloscope, vertical and horizontal sweep can not be triggered independently, instead the television "locks on" to synchronizing information encoded in the video signal.

One full vertical scan of the television screen defines a *field* containing 262.5 lines. (Vertical resolution may be doubled to 525 lines using interlaced fields, but 262.5 horizontal lines is sufficient for our purposes). The field is refreshed (redisplayed) at 60 Hz.

The video signal generated by the camera is a *composite* video signal. That is, the video data (gray levels), horizontal and vertical synchronization information is all encoded on a single wire. The television locks on to the horizontal and vertical sync information in the video signal. Horizontal sync defines the beginning of a new line, and vertical sync defines the beginning of a new field. The horizontal sweep rate is 60 fields per sec $\times$ 262.5 lines per field or 15.75 KHz. The horizontal sweep interval is divided into three portions as shown in Fig. 4: blanking, synchronization, and data. Each line lasts 63.5 μs (= 1.0/15.75KHz). Retrace occurs during the blanking interval, and the leftmost visible data occurs at the end of the blanking interval. The horizontal synchronization signal occurs approximately in the middle of the blanking interval. Note that the video signal is analog, not binary, with levels from white to black, to synch.

It is slightly more complicated to understand how vertical synchronization is generated. Generation of the vertical synchronization signal is shown in Fig. 5. Since the vertical scan is much slower than the horizontal scan, a longer vertical blanking interval is required to avoid seeing the retrace lines. The example in the figure indicates a blanking time of 21 horizontal periods. How many horizontal lines will be visible? Approximately 241 (= 262 -21) lines. The standard blanking interval is between 18 and 21 horizontal periods. It is ok if your design misses a few lines on the top or bottom of the screen. A wide VSYNC pulse triggers the vertical oscillator in the monitor to start another cycle, however, the horizontal oscillator must be kept synchronized during this time. Hence the serrations in the COMP_SYNCH.H signal. (See

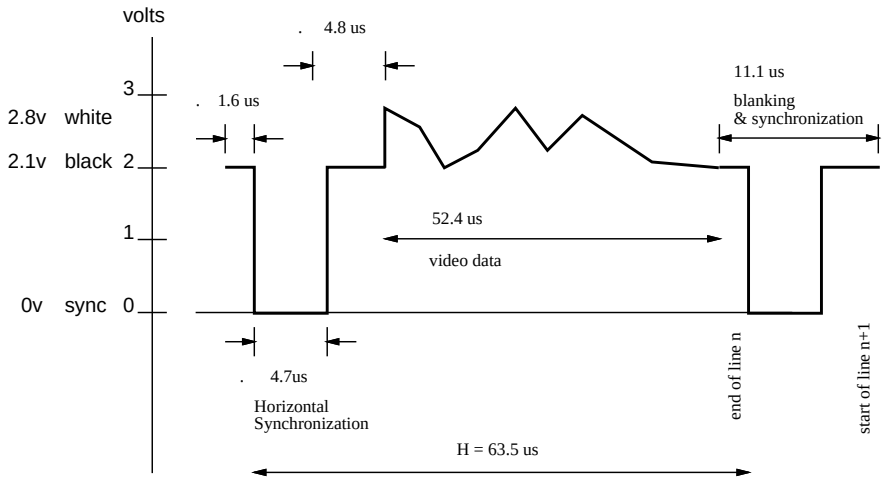


Figure 4: Approximate Horizontal Synchronization Timing. Note that only $52.43\mu s$ of the line is visible, the other $11.1\mu s$ is taken up by retrace and blanking time. Actual camera timings may vary.

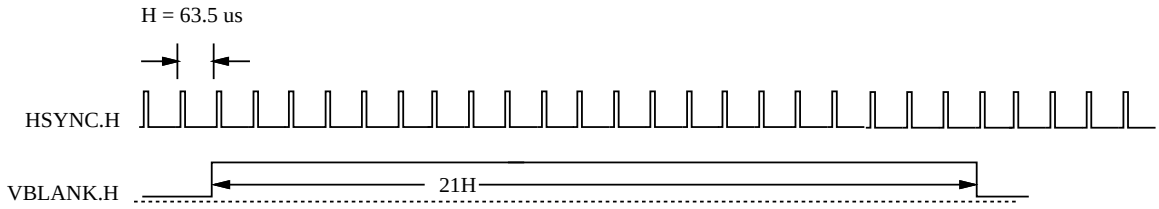


Figure 5: Approximate Vertical Synchronization Timing. (Please check your camera sync output to verify actual values.

LM1881 sync separator data sheet.)

Figuring out where the top and left side of the image is could be complicated. Fortunately, you will be using the LM1881 sync separator in checkpoint 3. It takes as input the composite video signal, and provides outputs of vertical sync and composite sync. So to find the beginning of the video frame, wait about 17 horizontal periods after the vertical sync signal goes low. Then begin reading in video data about 9.5 μs after the composite sync goes low. Read a line, then wait for the next line until you have read all the lines you need.

7 Project Deadlines

This section summarizes the functionality you need to demonstrate, as well as the steps in the design you need to follow. The number in parentheses is points out of 100 total for the project. Prelab assignments will only be checked off during your lab section during scheduled meeting times. You will only receive full credit for prelab if it is complete and checked off during your lab section. (Both partners need to show up to get individual prelab credit).

For checkpoints 1-4, there will be a formal design review with your TA during your scheduled lab section. You will need drawings and printouts on paper to hand in as part of a brief presentation on your progress and plans so far. Points are maximum, and will be based on completeness and quality of presentation as well as soundness of reasoning.

7.1 Checkpoint 1. Due in lab week of 18 Oct.

Prelab:

- (2) Detailed block diagram of data path and controller, (to level of Xilinx library components, counters, comparators, registers, etc).
- (2) State diagram for controller (initial attempt)
- (1) Project plan: division of labor, number of CLBs, testing strategy, milestones, etc.

Lab:

- (4) Demonstrate serial receiver.

7.2 Checkpoint 2. Due in lab week of 25 Oct.

Prelab:

- (3) State diagram for controller for project, complete.

Lab:

- (5) Demonstrate reading a small frame (10 by 14) from serial line, dumping RAM through PWM module to LED Array. (This is Mode 1.)

7.3 Checkpoint 3. Due in lab week of 1 Nov.

Prelab:

- (2) Schematics for data path

Lab:

- (2) Demonstrate functioning A/D converter and video waveforms.
- (2) Demonstrate video data storage in RAM.

7.4 Checkpoint 4. Due in lab week of 8 Nov.

Prelab:

- (2) Complete schematics for data path and controller. Simulation output for controller.

Lab:

- (5) Demonstrate capture a video frame in RAM from camera, display in grayscale on LED array.

7.5 Checkpoint 5. Due in lab week of 15 Nov.

Lab:

- (5) Demonstrate assigned feature for a video frame from assigned input (your Mode 2).
- Early Completion Bonus. +5 points if you demo the complete project during your scheduled lab section.

7.6 Project Week 11/22, 11/23, 11/24

- (45) Demo completed project, modes 1, 2, and movie playback, mode 3. Sign up for demo slots during your scheduled lab section. (Thursday 5-8 lab section only will sign up, and demo by 5 pm Wed. 11/24.)
- (20) Final Report is due Friday 12/3/98 in lab along with your unwrapped Xilinx kit and all components. Project grade will be zero and check cashed if Xilinx kit is not returned by 12/3/98.