Digital Electronics - Supervision 1

Nandor Licker <nl364@cl.cam.ac.uk>

October 2019

1 Warmup

Attemp Exercises 1 to 19 (inclusive) from the exercise sheet, individually. Feel free to discuss the rest of the exercise within your group: it is okay if you don't do it all before the first supervision, but you should at least try to understand what you should design. If you have any questions, do not hesitate to contact me.

2 Overview

In this exercise, you will design a VGA adapter for a 640 x 320 display refreshed at 60Hz, displaying 80×40 monochrome characters. The input is a RAM block storing the characters at specific addresses, along with a pixel clock running at 25.175Mhz. The outputs are the following 3 signals:

- Horizonal Sync (HSync): 0 or 5V
- Vertical Sync (VSync): 0 or 5V
- Intensity: 0 to 0.7V

VGA is quite an old protocol - it was originally designed to control the electron gun in a cathode ray tube (CRT). The electron ray scans the phosphor-coated glass line-by-line, from top to bottom. After finishing a line, the ray must travel back to the start of the previous one, reason why data transmission must wait between two lines. Similarly, after rendering a frame, the ray must travel back to the top-left corner of the screen from the bottom-left, introducing a delay between consecutive frames.

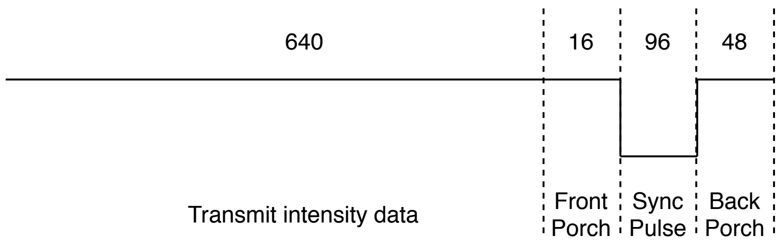

Figure 1: Horizontal Timing

Figure 1 illustrates the vertical timings: transmitting a whole line takes $640 + 16 + 96 + 48 = 800$ cycles of the pixel clock. The process is split into 4 phases:

- Data: for 640 cycles, HSync is high and intensity data is transmitted pixel-by-pixel
- Front Porch: HSync is high, no data is transmitted
- Sync Pulse: HSync is low, no data transmitted
- Back Porch: HSync is high, no data is transmitted

The sync pulse is required to synchronise the adapter with the display - it is usually high, except during the Sync Pulse phase when it is driven low.

Vertically, transmitting lines works similarly:

- Data: transmit 480 lines, VSync is high
- Front Porch: Wait for 10×800 cycles, VSync is high
- Sync Pulse: Drive VSync low for 2×800 cycles
- Back Porch: VSync is high again for 33×800 cycles

The VGA module you will design will consists of 3 main components: a horizontal timer, a vertical timer and pixel data generation, which will be discussed over the next sections.

3 Horizontal and Vertical Timing

The horizontal timer needs to count up to 800 and provide an overflow pulse to the vertical timer once it wraps to the next line. Its input is the pixel clock and its outputs are:

- HSync flag
- Overflow pulse to the vertical timer: once the counter reaches 800, it sets a pulse high for one cycle to enable the vertical timer to count one up.
- Pixel index to the character module
- Data enable flag: high during the first phase of the horizontal cycle, for 640 cycles

Design the horizontal timing module and draw its diagram.

1. Design a register capable of storing the pixel counter
2. Design a circuit to increment its value on every cycle
3. Design the combinatorial logic to output the HSync signal
4. Design the combinatorial logic to output the data enable flag
5. Wire the next pixel value back to the register using multiplexers
6. Design the combinatorial logic for the HSync pulse

Draw a high-level diagram connecting all these modules, then discuss in detail the combinatorial or sequential circuits inside of each module. After you are done with the horizontal module, move on to the design of the vertical one. Bear in mind the difference: the vertical counter is not incremented on every cycle, but each time the horizontal counter generates an overflow signal. Your design of the register storing the vertical number should reflect this. Make sure to avoid hazards: do not mix combinatorial logic with the clock signal, use registers which have a write-enable bit.

4 Pixel Data

The horizontal and vertical counters provide this module with 4 inputs:

- Horizontal index
- Vertical index
- Horizontal enable
- Vertical enable

Furthermore, character data is provided by a suitably sized RAM. Your goal is to generate pixel data from character data: this is achieved through the use of an intermediary ROM. The ROM encodes the bitmaps which should be displayed for each character. We assume that the characters are represented using 8 bits, the RAM and the ROM are byte addressable and that the displayed glyphs are 8×8 pixels in size. Design the pixel generation logic, stepping through the following questions:

- Given the vertical and horizontal indices, which pixel of which character do you need to display?
- How do you lay out the characters in the character RAM? At what address is it most effective to store each charater?
- How do you lay out the glyphs in the character ROM? What is the size of the ROM?
- The ROM is byte-addressable, how do you select the pixel you need to display?

Draw a high-level diagram, showing how you generate addresses for the RAM and the ROM, along with the logic to select the final pixel. Describe the combinatorial components in more detail. How fast do the RAM and the ROM have to be?

5 Digital to Analog

Given a pixel intensity, you need to drive the line carrying the signal to the display. Digital logic usually operates at 0V or 5V, but VGA requires 0.7V. Show how you generate this signal, along with any required amplification.

6 It all fits!

Draw a diagram indicating the connections between the modules designed previously.

7 Extension to Colour

To display in colour, your module now needs to drive 3 signals - R, G and B. Discuss how you would colour each character and its background differently - how much RAM would you need for this task?