Architecture, réseaux et système I
Homework

Deadline 24 October 2008

Andreea Chis, Matthieu Gallet, Bogdan Pasca

October 16, 2008

Text-mode display driver

Problem statement

Design the architecture for a text-mode display driver having the black-box model shown in figure 1

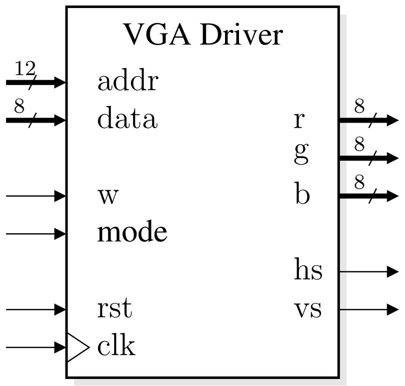

Figure 1: Top level blackbox view of the final circuit

Problem Analysis

The VGA screen can be seen as a matrix of pixels. Displaying a character on the screen translates to lighting certain pixels of this matrix. One solution in designing our display driver would be to give the user access to independent pixels of the matrix. This would facilitate the display of images but would render overwhelming the task of displaying a text.

The role of the **text-mode display driver** is to offer the user a high-level **interface** for **displaying characters** on a **VGA display**. The previous mentioned pixel matrix is transformed into a character matrix. This matrix holds the codes of the characters to be displayed on the screen. The user has write access to all elements of this matrix. The information about how to draw each character is internally stored in a ROM (read only memory). The character codes stored in the matrix are used to address this memory in order to retrieve the character information.

On CRT-based VGA displays the image is formed by scanning the viewing screen with an electron beam in a **sequence of horizontal lines**. *[The amplitude-modulated electron beam (one for each RGB channel) hits a phosphor coated screen. The phosphor surface glows brightly at the moment of the impact*

[0]For more information and resources visit http://perso.ens-lyon.fr/bogdan.pasca/teaching.html or email Bogdan.Pasca@ens-lyon.fr

with the beam and continues to glow for several hundred microseconds. The brightness of the impact is directly proportional to the amplitude of the beam.]

The electron beam moves on the display surface horizontally from **left to right** and vertically from **top to bottom**.

[Analogy: Think of an old typewriter. The letters appeared on the first row of the page from left to right. When the cursor was in the rightmost position, it had to be moved to the next row in the leftmost position. These transitions are called in display terminology ***blanking periods****. When the end of a page was reached, the cursor had to be reseted to the top leftmost position. This is also called a blanking period.]*

Information is displayed only when the cursor moves in forward direction. Consequently, much of the time is lost in blank periods, when beam is reset and stabilized to begin a new pass. The VGA controller generates two synchronizing signals - Horizontal Sync (hs) and Vertical Sync (vs) to synchronize the electron beam with video data delivery.

Figure 2 shows the synchronization timings for a VGA display at a resolution of 640x480 pixels, using a 25MHz pixel clock and having a 60Hz vertical refresh rate. The **pixel clock** defines the **time available to display one pixel of information**. These are the timings needed to be fed to the monitor in order to synchronize the electron beam with the data sent to the display.

As figure 2 shows, in order to display a complete horizontal line having 640 pixels, the electron beam needs to spend 800 pixel clock cycles. The additional number of cycles (800-640=160) is spent on reseting and stabilizing the electron beam in order to start a new line. The top part of figure 2 shows the trajectory of the beam in displaying two consecutive horizontal lines of pixels. A low **hs** pulse having a length of 96clk cycles marks the beginning of a horizontal line. This is followed by a back porch period of 48 clock cycles. Then for 640 clock cycles pixel data is sent to the display. Next, for 16 clock cycles no data is sent and then a new line follows with a negative hs pulse. This is all summed up at the bottom of figure 2.

The table below synthetises the correct values for synchronizing a display at a resolution of 640x480 pixels, using a 25MHz pixel clock and having a refresh rate of 60Hz. The notations are the same those for figure 3.

Symbol	Parameter	Vertical sync			Horizontal Sync	
		Time	Clock	Lines	Time	Clocks
T_s	Synchronization pulse time	16.7 ms	416,800	521	32 us	800
T_{disp}	Display Time	15.36 ms	384,000	480	25.6 us	640
T_{pw}	Pulse Width	64 us	1,600	2	3.84 us	96
T_{fp}	front porch width	320 us	800	10	640 ns	16
T_{bp}	back porch width	928 us	23,200	29	1.92 us	46

Note that modern LCD monitors use the same protocol to display images, although no electron beam is present.

Requests

1. Draw the architecture and automaton (finite state machine) for synchronizing with a VGA display at a resolution of **640x480** pixels, having a **25MHz** pixel clock. The output color to the display can be a constant. The black-box model of the circuit is presented in figure 4. You are provided with a clock signal having a frequency of 100 MHz. State your design decisions. *[Remember that the* ***pixel clock*** *(25MHz) must be obtained from the* ***master clock*** *(100MHz) and that all synchronization values for* ***hs*** *and* ***vs*** *are in terms of this pixel clock.]*

2. Modify the above architecture in order that the display goes in cycle from white to black and back to white through all shades of gray. State forward your design choices. *[Remember that the different shadows of gray are obtained when r=g=b].*

3. You are in the possesion of a **fast synchronous memory** having the capacity of 512KBytes and an organization of 512K x 8 bits (see figure 5. The memory stores a monochrome image with the color information of each pixel being stored on the least significant bit (LSB) of the corresponding

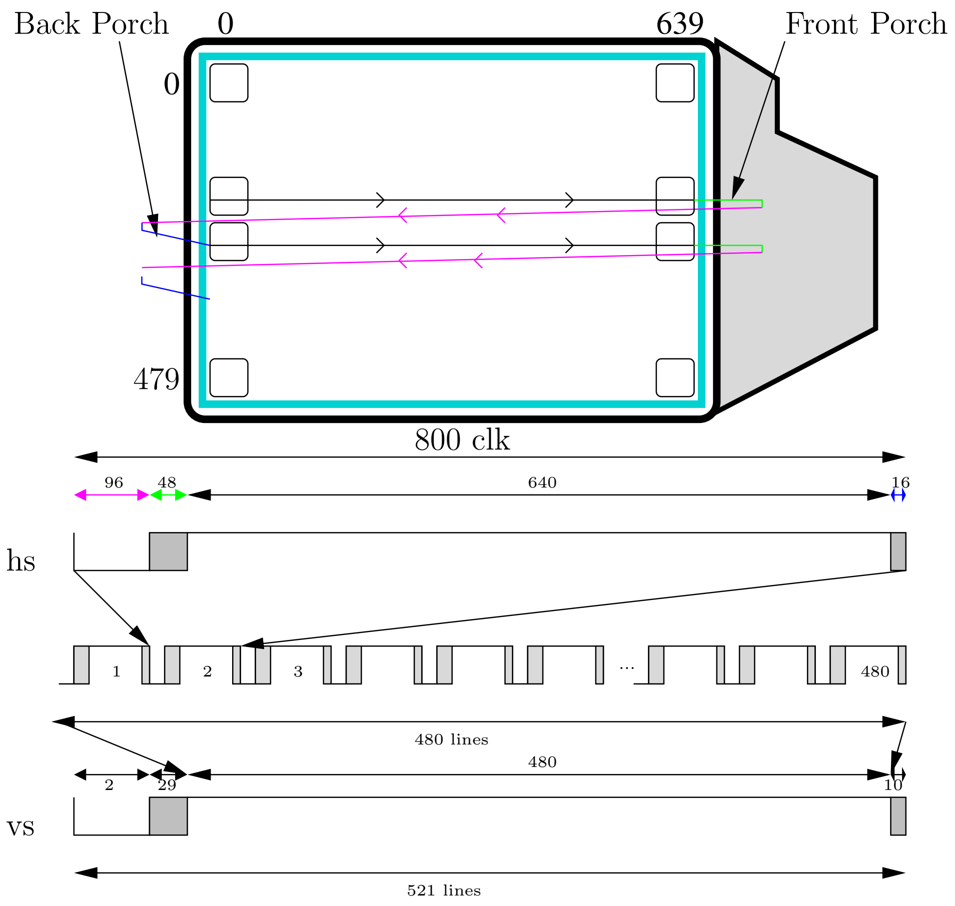

Figure 2: VGA display timing model for 640x480 resolution

addresses. Consecutive addresses store consecutive pixels from a line. For example, the pixel at coordinates (0,0) is stored at address 0, (1,0) at address 1, (0,1) at address 640.

Modify the architecture above so that the image from the memory is displayed on the screen. *[Remember what you have to do in order to address the memory and in order to set the RGB channels correctly].* Draw the corresponding automaton and argument your decisions.

4. We are now in the case when each pixel of the image contains 2 bits of information for each color channel (6 in total) + a 2 bit alpha channel. The default background of the image is white. The alpha channel values range from 3 to 0. The value 0 of the alpha channel for a pixel is means that the pixel takes the color of the background(in our case white). A value of 3 means that the color is the one given by the 6 color bits r,g,b. Values ranging from 2 to 1 produce more and more transparency. It is your design decision to find a way that values 2 and 1 for alpha produce more transparency *[Remember that the background is white (11 11 11) for RGB and that more*

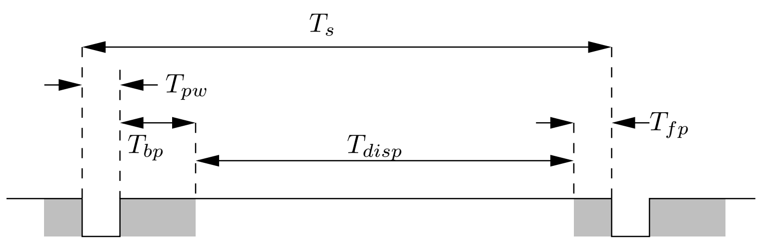

Figure 3: VGA display timing model for 640x480 resolution - detailed

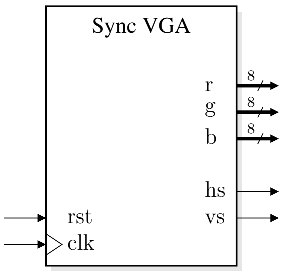

Figure 4: Black-box model for architecture at request 1

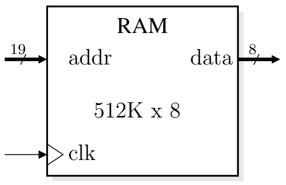

Figure 5: Fast Synch RAM used as ROM

transparency means moving (r,g,b) towards (11,11,11)]. The content of the RAM memory at an address containing the information of a pixel is organized as presented in figure 6. The a - represent the bits of the alpha channel, r-red, g-green, b-blue.

7							0
a	a	r	r	g	g	b	b

Figure 6: Information organization of one address of the RAM

Draw the architecture and argument all your decisions.

5. Starting from the above architecture, imagine that at each address, for the corresponding pixel, the RAM memory does not contain anymore the color information and the alpha channel, but an 8-bit address to a asynchronous ROM memory where this information is found. The ROM has 256 address lines and an organization of 256 x 8 (see figure 7). Introduce this new ROM into the architecture. Explain your decisions.

6. The screen is divided now into tiles of 8x8 pixels, (80 x 60 tiles). The RAM memory has now a capacity of 5KB and an organization of 5K x 8 bits. The information stored in the RAM for each tile is an 8 bit address. This address is used to access to asynchronous ROM presented at the previous point in order to obtain the color information about the tile. Draw the architecture which displays the tiled screen, each tile having the corresponding color given by the ROM information. Argument the decisions you take.

7. We now make available 8 ROMs, each having the capacity 256Bytes and organization 256x8bits (see figure 7). The screen is still divided into 8x8 pixel tiles and the RAM address still contains 8-bit addresses. We return to the monochrome world where each pixel has one bit of information.

   Let us take a tile k, T_{R_k,C_K}. The tile is defined by its row and column, R_k and C_k respectively. Let a matrix of 8x8 elements represent the 8x8 pixel tile.

   Let us denote the ROM memories by $ROM_h h \in N$, $0 \leq h \leq 7$. The information for the pixels of row j from tile k is found in memory ROM_j at the address given by $RAM(80*R_k+C_K)$, that is, the content of the RAM at that address. Pixel i from row j is found at $ROM(RAM(80*R_k+C_K))[i]$ in ROM_j.

   In other words, the RAM content are still addresses. The information for the 8 rows of the tile is found in the 8 ROMs. The pixels of a row are mapped into the ROM data (see figure 8).

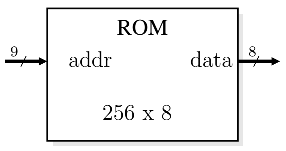

Figure 7: A 256x8 ROM

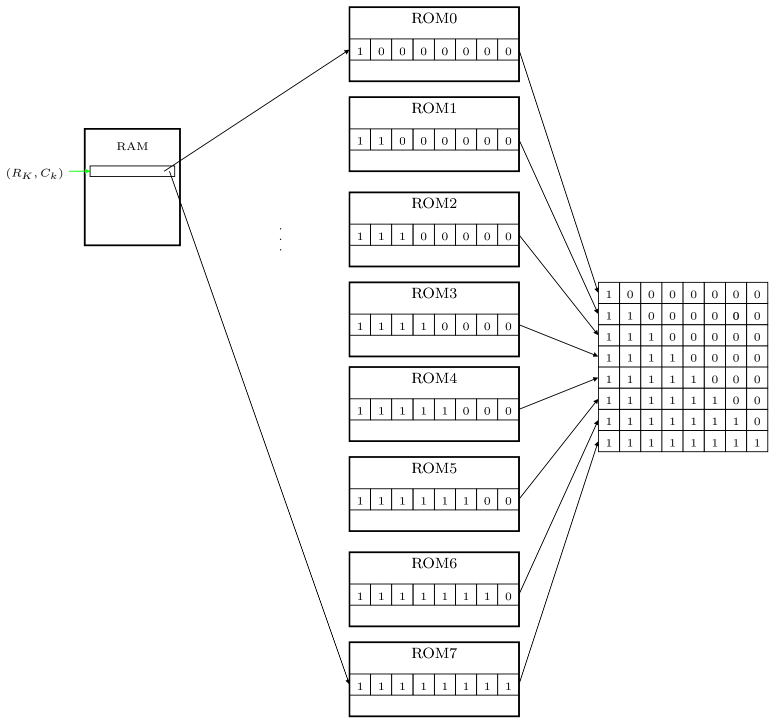

Figure 8: RAM addresses ROMs which contain pixel data of tile

Design the architecture and explain your decisions.

8. The architecture at the previous point is able to write predefined tiles from the 8 ROM memories. Instead of the 8 ROMs we now have one ROM with the capacity 2KBytes, and organization of 2K x 8 bits. The information for a tile which was previously found in the 8 ROMs combined is now found in consecutive rows in the new one (see figure 9). Consider that the tiles represent characters as shown for letter A in fig 9. This new architecture is able to draw characters on the display.

   Draw the architecture and argument your choices.

9. At the previous point we were able to display characters who's codes were already stored in the RAM memory. We desire to take this design a step further.

   Let us consider the memory module shown in figure 10. The significance of the I/O ports is:

   $\overline{WE}$ - gives write access to the SRAM. The overline indicates that the signal is active low. In other words, when this signal has a value of 0 write access is permitted to the memory.

   $\overline{OE}$ - gives read access to the SRAM. The overline indicates that the signal is active low. In other words, when this signal has a value of 0 read access is given to the memory. When this line is 1, the data line remains in high impedance.

   data - bidirectional data line. When $\overline{WE}$ is active this is an output. When $\overline{OE}$ is active this is an input.

   $\overline{OE}$ and $\overline{WE}$ must never simultaneously be active as this will cause damage to the SRAM.

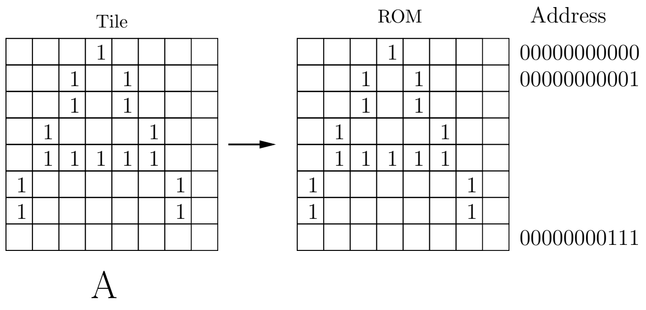

Figure 9: ROM content

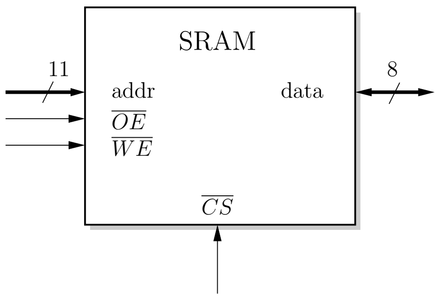

Figure 10: SRAM layout

The signal timings for writing the SRAM are given below and in figure 11

- put CS on 0 (the memory has read/write access)
- put address on address lines
- put data on data line
- keep $\bar{\text{O}}$E on high
- put $\bar{\text{W}}$E low
- clock validate

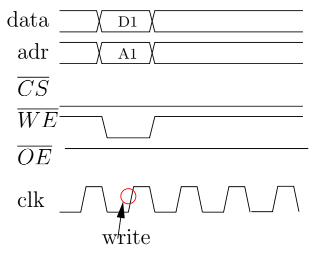

Figure 11: Writing timings for SRAM

Augment the architecture designed at the previous point so that the data from the SRAM can be modified dynamically, without stopping access to the SRAM for the display. *[Hint: think at the time when the electron beam is not active (blanking periods) to give writing access to the memory. Think of a queue mechanism.]*

Explain your design decisions.

Expert points:

1. Use a **mode** input to differentiate between 2 operating modes of the display (80x25 characters and 40x15 characters) *[Special care must be taken when scaling the letters]*. Design the architecture and explain your decisions.

2. Use additional memory to store color information for each letter. Design the architecture which is able to display colored text. You have full control over the components you use. Design the architecture and explain your decisions.

The requirement are set in an order which is constructive and leads to solving the more difficult points. If at some point you feel that the requirement is not clear you can e-mail Bogdan.Pasca@ens-lyon.fr for details. My suggestion is to start incrementally with the first points and then build up to the final architecture.

Please make sure when drawing the architectures to to specify the direction of the dataflow by placing direction arrows on the wires.