ECE 243S - Computer Organization
March 2025

Introduction to the VGA Controller - The Frame Buffer and Character Buffer

The Frame Buffer

The DE1-Soc has a video-out port with a VGA controller that can be connected to a standard VGA monitor. The VGA controller supports a screen resolution of 640 $\times$ 480 pixels, but we'll be using it at a lower resolution. The image displayed by the VGA controller is in a region of memory called the *frame buffer* (which the intel documentation refers to as a *pixel buffer*), which can be set by the programmer. This frame buffer for the video-out port holds the data (color) for each pixel that is displayed by the VGA controller. As illustrated in Figure 1, the frame buffer provides an image resolution of 320 $\times$ 240 pixels, with the coordinate 0,0 being at the top-left corner of the image. Since the VGA controller supports the screen resolution of 640 $\times$ 480, the controller replicates each pixel value from the frame buffer in both the *x* and *y* dimensions when displaying an image on the VGA screen.

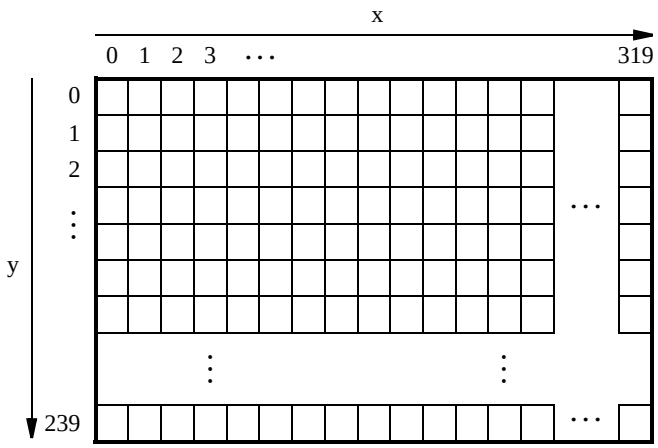


Figure 1: VGA screen coordinates.

Figure 2*a* shows that each pixel color is represented as a 16-bit halfword (two bytes) with five bits for the blue and red components, and six bits for green. As depicted in part *b* of Figure 2, pixels are addressed in the frame buffer by using the combination of a *base* address and an *x,y* offset. The default address of the start of the frame buffer is 0x08000000. This address corresponds to the beginning of a memory block that is located inside the FPGA chip in the computer system. Using the addressing scheme in Figure 2*b*, the pixel at location 0,0 has the address 0x08000000, the pixel 1,0 has the address *base* + $(00000000\ 000000001\ 0)_2$ = 0x08000002, the pixel 0,1 has the address *base* + $(00000001\ 000000000\ 0)_2$ = 0x08000400, and the pixel at location 319,239 has the address *base* + $(11101111\ 100111111\ 0)_2$ = 0x0803BE7E.

You can create an image on the screen by writing color values into the pixel addresses as described above. A dedicated *frame buffer controller* (which is hardware in the FPGA) reads this pixel data from the memory and sends it to the VGA display. The controller reads the pixel data in sequential order, starting with the pixel data for the upper-left corner of the VGA screen and proceeding to read the whole buffer until it reaches the lower-right corner. This process is then repeated, continuously. You can modify the pixel data at any time, by writing to the pixel addresses. Writes to the frame buffer are automatically interleaved in the hardware with the read operations that are performed by the frame buffer controller.

It is also possible to prepare a new image for the VGA display without changing the content of the frame buffer, by using the concept of *double-buffering*. In this scheme two frame buffers are involved, called the *front* and *back* buffers, as described below.

Double Buffering

As mentioned above, a frame buffer controller reads data out of the frame buffer so that it can be displayed on the VGA screen. This frame buffer controller includes a programming interface in the form of a set of registers, as illustrated in Figure 3. The register at address 0xFF203020 is called the *Buffer* register, and the register at address 0xFF203024 is the *Backbuffer* register. Each of these registers stores the starting address of a frame buffer. The Buffer register holds the address of the frame buffer that is displayed on the VGA screen. As mentioned above, in the default configuration of the computer systems the Buffer register is set to the address 0x08000000, which points to the start of the FPGA on-chip memory. The default value of the Backbuffer register is also 0x08000000, which means that there is only one frame buffer. But software can modify the address stored in the Backbuffer register, thereby creating a second frame buffer. An image can be drawn into this second buffer by writing to its pixel addresses. This image is not displayed on the VGA monitor until a frame buffer *swap* is performed, as explained below.

The frame buffer can be located in the SDRAM memory in the DE1-SoC Computer, which has the base address 0x00000000 (shared with your binary code). You should place the frame buffer in the data segment to avoid conflicts with your binary code. For example, if you declare a global array Buffer[ResolutionX][ResolutionY + Padding] in your C code, the compiler will place it in the data segment. Due to the default addressing scheme of the frame buffer controller, each row of the frame buffer must be aligned to a 512-column boundary. For the purpose of double buffering, it is more convenient to declare both front and back buffers in the data segment.

A frame buffer swap is caused by writing the value 1 to the Buffer register. This write operation does not directly modify the content of the Buffer register, but instead causes the contents of the Buffer and Backbuffer registers to be swapped. The swap operation does not happen right away; it occurs at the end of a VGA screen-drawing cycle, after the last pixel in the bottom-right corner has been displayed. This time instance is referred to as the *vertical synchronization* time, and occurs every 1/60 seconds. Software can poll the value of the S bit in the *Status* register, at address 0xFF20302C, to see when the vertical synchronization has happened. Writing the value 1 into the Buffer register causes S to be set to 1. Then, when the swap of the Buffer and Backbuffer registers has been completed S is reset back to 0. The *Status* register contains additional bits of information, shown in Figure 3. The m and n bits specify the number of y and x VGA address bits, respectively. The *BS* bits indicate pixel-size; for a pixel size of two bytes, this field is set to 15. Also, the programming interface includes a *Resolution* register, shown in the figure, that contains the X and Y resolution of the frame buffer(s).

In a typical application the frame buffer controller is used as follows. While the image contained in the frame buffer that is pointed to by the Buffer register is being displayed, a new image is drawn into the frame buffer pointed to by the Backbuffer register. When this new image is ready to be displayed, a frame buffer swap is

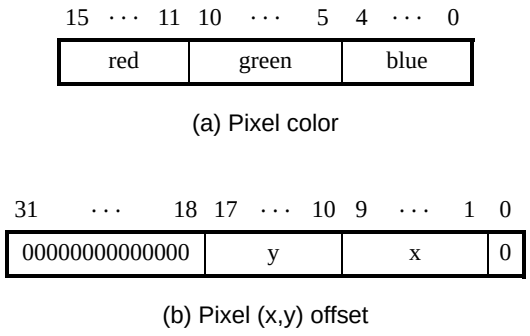


Figure 2: Pixel values and addresses.

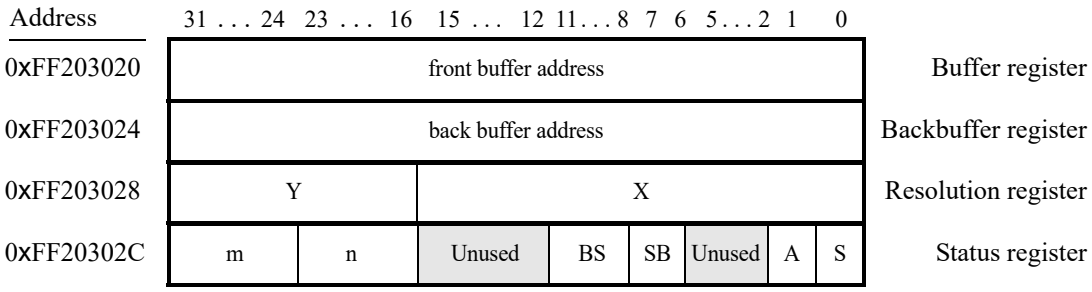
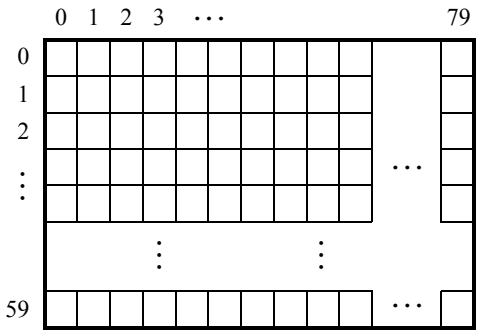


Figure 3: Frame buffer controller registers.

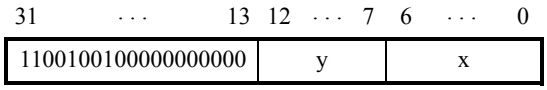
performed. Then, the frame buffer that is now pointed to by the Backbuffer register, which was already displayed, is cleared and the next new image is drawn. In this way, the next image to be displayed is always drawn into the "back" frame buffer, and the "front" and "back" buffer pointers are swapped when the new image is ready to be displayed. Each time a swap is performed software has to synchronize with the VGA controller by waiting until the S bit in the Status register becomes 0.

Appendix

As mentioned earlier, on a DE1-SoC board the image displayed by the VGA controller is based on the data in the frame buffer, which displays graphics. But there is also a *character* buffer, which makes it easy to display text. This information may come in handy for the project in this course. The character buffer is stored in FPGA on-chip memory in the DE1-SoC Computer. Figure 4*a* depicts the character buffer for the VGA display, which has a resolution of 80 × 60 characters. Each character occupies an 8 × 8 block of pixels on the screen. Characters are stored in each of the locations shown in Figure 4*a* using their ASCII codes; when you store an ASCII character into the buffer, a corresponding pattern of pixels is automatically generated and displayed using a built-in font. Part *b* of Figure 4 shows that characters are addressed in the memory by using the combination of a *base* address, which has the value $(09000000)_{16}$, and an *x,y* offset. Using this scheme, the character at coordinates $(0, 0)$ has the address $(09000000)_{16}$, $(1, 0)$ has the address *base* + $(000000\ 0000001)_2$ = $(09000001)_{16}$, $(0, 1)$ has the address *base* + $(000001\ 0000000)_2$ = $(09000080)_{16}$, and the character at location $(79, 59)$ has the address *base* + $(111011\ 1001111)_2$ = $(09001DCF)_{16}$.



(a) Character buffer coordinates



(b) Character buffer addresses

Figure 4: Character buffer coordinates and addresses.

The character buffer has an associated controller, with a register interface like the one shown in Figure 3. This controller has the base address 0xFF203030. You can read from the *Resolution* register to obtain the number of text columns (X) and rows (Y) on the screen. For the VGA screen, the values will be 80 columns × 60 rows.