Video Output Interface

The Altera University Program provides a number of hardware controllers, called cores, to control the VGA Digital-to-Analog Converter (DAC) and display images on a screen. These include a *VGA Pixel Buffer*, a *VGA Character Buffer*, and a *VGA Controller* circuit, which are used together with the a memory module and the corresponding memory controller to allow programs executed by the Nios II processor to generate images for display on the screen. If both the *Pixel* and *Character Buffers* are being used in the same system, the *Alpha Blending* core must be used to combine the two video streams.

VGA Display Basics

A VGA video signal contains 5 active signals [1]. Two signals compatible with TTL logic levels, horizontal sync and vertical sync, are used for synchronization of the video. Three analog signals with 0.7 to 1.0-Volt peak-to-peak levels are used to control the color. The color signals are Red, Green, and Blue. They are often collectively referred to as the RGB signals. By changing the analog levels of the three RGB signals all other colors are produced.

The screen refresh process seen in Fig. 1 begins in the top left corner and paints 1 pixel at a time from left to right. At the end of the first row, the row increments and the column address is reset to the first column. Each row is painted until all pixels have been displayed. Once the entire screen has been painted, the refresh process begins again. The video signal paints or refreshes the image using the following process. The vertical sync signal, as shown in Fig. 2 tells the monitor to start displaying a new image or frame, and the monitor starts in the upper left corner with pixel (0, 0). The horizontal sync signal, as shown in Fig. 3, tells the monitor to refresh another row of 640 pixels. After 480 rows of pixels are refreshed with 480 horizontal sync signals, a vertical sync signal resets the monitor to the upper left comer and the process continues. During the time when pixel data is not being displayed and the beam is returning to the left column to start another horizontal scan, the RGB signals should all be set to the color black (all zeros).

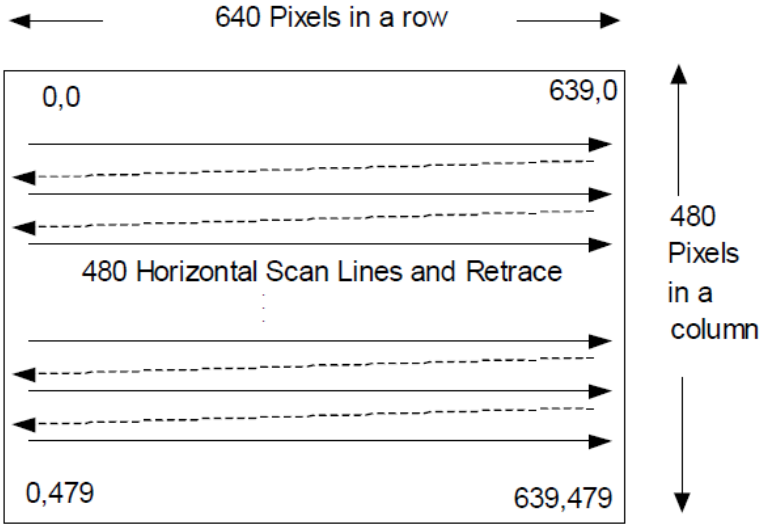


Fig. 1. VGA pixel layout (640x480).

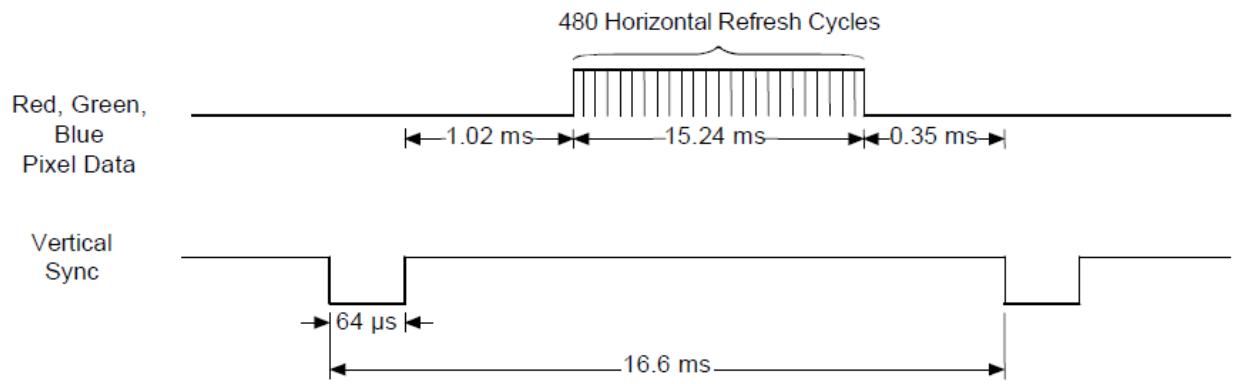


Fig. 2. Timing for vertical sync signal.

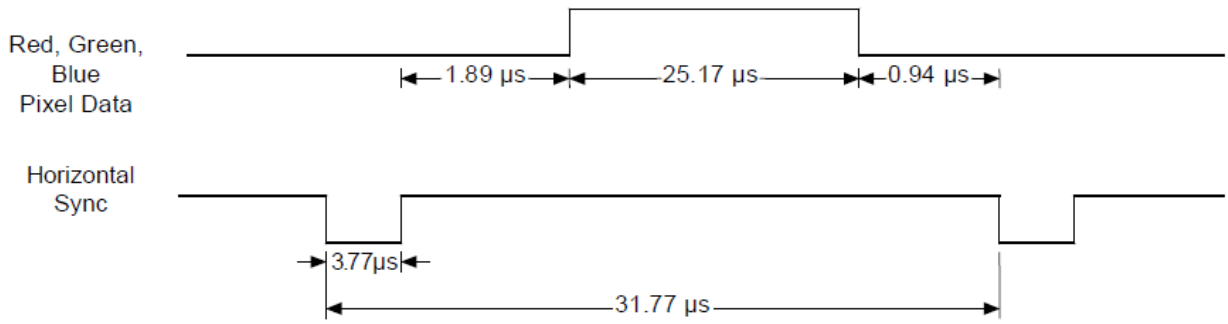


Fig. 3. Timing for horizontal sync signal.

Pixel Buffer

Altera provides a VGA controller that can access pixel colors from a memory chip and send color information along with the horizontal sync and vertical sync signal to a VGA DAC. The *Pixel Buffer* sends pixel color values from a memory buffer to the VGA Controller, via an Avalon Streaming Interface [2]. The starting address of the memory buffer is user selectable. Using the default settings, pixel values consist of 16 bits, and the memory buffer is accessed using half-word operations. The pixel buffer converts the 16-bit pixels to the 30-bit pixels required by the VGA Controller, and scales the resolution where appropriate. The pixel buffer can be configured to provide different resolutions. Using the default settings, the resolution is 320x240 pixels for the VGA DAC.

The *VGA pixel buffer* gives the size of the screen and the **location in the SRAM memory** where an image to be displayed is stored. To display an image on the screen, the *VGA pixel buffer* retrieves it from memory and sends it to the *Alpha Blending* core which combines two video streams into one. For this lab, the *Alpha Blending* core is used to combine video streams from *VGA pixel* and *VGA character* buffers. Fig. 4 shows the block diagram for the *Alpha Blending* core.

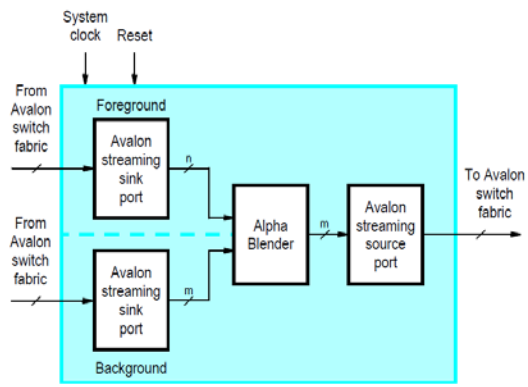


Fig. 4. Block diagram for the Alpha Blending core.

The *Alpha Blending* core sends the combined video stream to the *VGA controller*. The *VGA controller* then uses the VGA DAC to send the image data across the VGA cable to the screen.

An image consists of a rectangular array of picture elements, called pixels. Each pixel appears as a dot on the screen, and the entire screen consists of 320 columns by 240 rows of pixels, as illustrated in Fig. 5. Pixels are arranged in a rectangular grid, with the coordinate (0; 0) at the top-left corner of the screen.

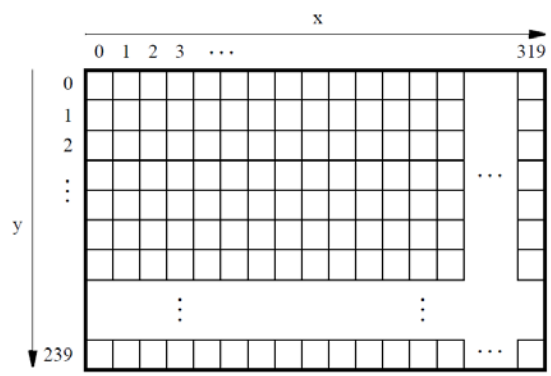


Fig. 5. Pixel array.

The address of each pixel is formatted by adding an offset to the base address. The supported addressing modes are called consecutive mode and X-Y mode.

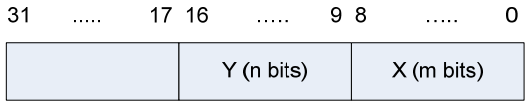
- ***Consecutive mode***— the pixel addresses are consecutively laid out in the addressable space. For example, for 320x240 resolution the pixel at screen coordinate (0, 0) is at the offset 0x0000, (0, 1) is at offset 0x0001, ... (0, 319) is at offset 0x13F, (1, 0) is at offset 0x140, and so on.
- ***X-Y mode***—the address contains x and y coordinates. The format of the address is shown in Fig. 6. The values of *m* and *n*, shown in the figure, are related to the *VGA Core's* resolution, as follows:

$$m = ceil(\log_2 X)$$
$$n = ceil(\log_2 Y)$$

where X and Y are the resolution in the X, Y directions, respectively. For example, for 320x240 resolution we have

$$m = ceil(\log_2 320) = 9$$

$$n = ceil(\log_2 240) = 8$$



$$Pixel\ Address\ (x, y) = Base\ Address\ +\ offset$$
$$= Base\ Address\ +\ (y\ <<\ 9\ +\ x)$$

Fig. 6. X-Y address format for the 320x240 resolution.

The color of a pixel is a combination of three primary colors: red, green and blue. By varying the intensity of each primary color, any other color can be created. We use a 16-bit half-word to represent the color of a pixel. The five most-significant and least-significant bits in this half-word represent the intensity of the red and blue components, respectively, while the remaining six bits represent the intensity of the green color component, as shown in Fig. 7. For example, a red color would be represented by a value 0xF800, a purple color by a value 0xF81F, white by 0xFFFF, and gray by 0x8410.

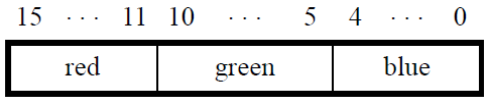


Fig. 7. Pixel Buffer's data format for the 16-bit color mode.

The *VGA pixel buffer* module contains memory-mapped registers that are used to access the VGA pixel buffer information and control its operation. These registers are listed in Fig. 8. The *Buffer* and *Back buffer* registers store the locations in the memory where two image buffers are located. The first buffer, called the *front buffer*, is the memory where the image currently visible on the screen is stored. Initially, both buffers can point to the same image. If the computer takes too much time to render a new image on screen which may appear to flicker, the *Back buffer* can be used for processing and the *front buffer* can be used for displaying. This technique is called *double buffering*. To enable double buffering, we need to separate the *front* and the *back buffers*. For example, use half of the SRAM memory for the *front buffer*, and the other half for the *back buffer*. The *Back buffer* register allows the start address of the memory buffer to be changed under program control. To change the memory buffer address:

- The desired new address is first written into the *Back buffer* register.
- Then, a second write operation is performed using the address of the read-only *Buffer* register. The value of the data provided in this second write operation is not used by the pixel buffer instead, it interprets a write to the *Buffer* register as a request to swap the contents of the *Buffer* and *Back buffer* registers. The swap does not occur immediately. Instead, the swap is done after the *Pixel Buffer* reaches the last pixel value associated with the screen currently being drawn by the *VGA controller*. While this screen is not yet finished, the bit *S* of the status register is set to 1. After the current screen is finished, the swap is performed and bit *S* is set to 0.

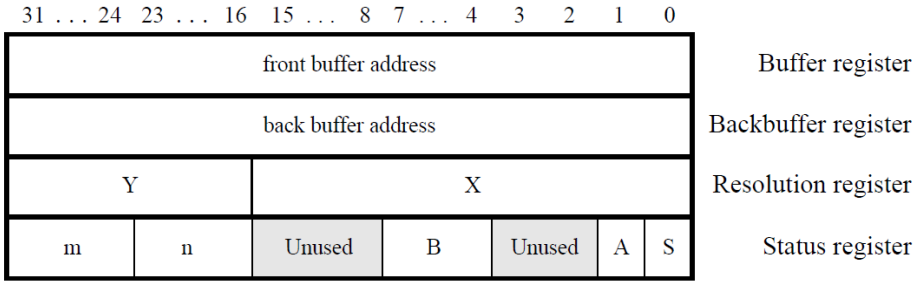


Fig. 8. VGA pixel buffer memory-mapped registers.

The *Resolution* register provides the *X* resolution of the screen in bits 15-0, and the *Y* resolution in bits 31-16. Finally, the status register provides information for the *Pixel Buffer*,

- *m*: width of the X coordinate (bits 31-24).
- *n*: width of the Y coordinate (bits 23-16).
- *B*: number of bytes of color: 1 (grayscale), 2 (16-bit color), 3 (24-bit color) or 4 (30-bit color).
- *A*: address mode: 0 (X,Y), or 1 (consecutive).
- *S*: swap: 0 when swap is done, else 1.

Character Buffer

The *Character Buffer's* unit of operation is a character. A device can send ASCII character codes to the *Character Buffer*'s Avalon interface *avalon_char_buffer_slave*. The *Character Buffer* handles the conversion of characters to pixels, and sends them to the *VGA Core* via an Avalon Streaming Interface. Upon initialization or reset, the *Character Buffer* sets all the characters to "space", so no characters will be displayed. This "clear screen" operation can take up to 5000 clock cycles to finish. In the *Character Buffer*, the resolution is defined by the number of characters per line and the number of lines per screen. The *Character Buffer* currently supports only one resolution per output device, where each character occupies an 8x8 VGA pixel group. Therefore, 80 characters can be displayed per line with 60 lines, since the *Character Buffer* is streaming the pixel values to the *VGA Core*, which in turn has a resolution of 640x480. The *Character Buffer* currently supports only one color option, which is that characters are drawn in white with a transparent background.

The core stores the characters in its **on-chip memory**.

When the *Character Buffer* is instantiated in an SOPC Builder system, it is assigned a base address, as a memory-mapped device. Each character in the buffer then has a unique address in this addressable space. The coordinate system of the Character Buffer is illustrated in Fig. 9. As the figure indicates, each character location is identified by an (x, y) coordinate, with (0, 0) being the top-left corner of the screen. The address in the buffer of each location is formed by combining the (x, y) coordinates as shown in Fig. 10, and t**hen adding this value to the base address**. All data for the *Character Buffer* is **character (8-bit) addressable**. The supported addressing mode is called the X-Y mode.

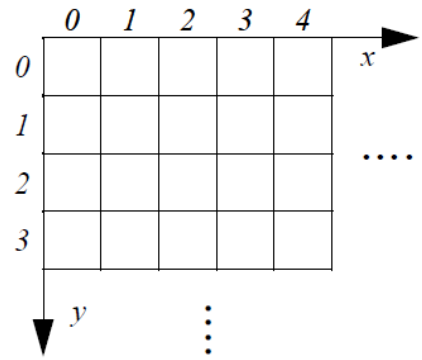


Fig. 9. Character coordinate system.

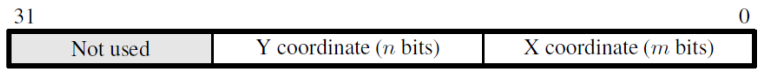


Fig. 10. X-Y address format.

The values of m and n are related to the VGA Core's resolution as follows:

$$m = ceil(\log_2 X)$$
$$n = ceil(\log_2 Y)$$

where X and Y are the resolution in the X, Y directions, respectively.

For example, for 80x60 resolution we have

$$m = ceil(\log_2 80) = 7$$
$$n = ceil(\log_2 60) = 6$$

and this resolution leads to the address format shown in Fig. 11.

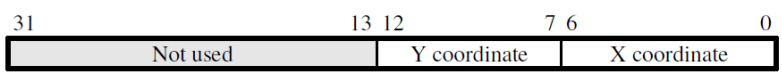


Fig. 11. X-Y address format for the 80x60 resolution.

Device drivers control and communicate with the *Character Buffer* through two Avalon memory mapped interface, named *avalon_char_control_slave* and *avalon_char_buffer_slave*. The *avalon_char_buffer_slave* interface has a one byte data width for ASCII characters and is addressed as described in the *Character Buffer*'s Address Format section above. The *avalon_char_control_slave* interface consists of the two registers shown in Table 1. The *Control* register provides the ability to clear the screen by using the *R* bit, which is **bit 16** of this register. The *R* bit remains set to 1 until all characters have been cleared, and then *R* is set to 0. The *Resolution* register, which is read-only, provides two values: the number of characters per line, in **bits 15-0**, and the number of lines per screen, in **bits 31-16**.

Table 1. Character Buffer register map					
Offset in bytes	**Register Name**	**R/W**	**Bit Description**		
			31…17	**16**	**15…0**
0	`Control`	RW	*(1)*	R	*(1)*
4	`Resolution`	R	Lines		Chars

Notes on Table 1:

(1) Reserved. Read values are undefined. Write zero.

References

[1] Chapter 10 in *Rapid Prototyping of Digital Design book, SOPC Edition*, 2008.

[2] Altera, "*Video Out IP Cores for Altera DE Boards,*" 2009.