VGA Generator for the XSA Boards

Summary

This application note discusses the timing for the video signals that drive a VGA monitor and describes a simple circuit that will let you display an image stored in the SDRAM on the XSA Board.

VGA Generator Features

- Flexible timing for the horizontal and vertical sync signals.

- Adjustable width for the red, green and blue output signals.

- 255-word pixel buffer for storing pixels.

- Selectable pixel width so each word of memory can hold 1, 2, 4, 8 or 16 pixels.

Principles of VGA Video

VGA Color Signals

There are three signals -- red, green, and blue -- that send color information to a VGA monitor. These three signals each drive an electron gun that emits electrons which paint one primary color at a point on the monitor screen. Signal levels between 0 (completely dark) and 0.7 V (maximum brightness) control the intensity of each color component, which combine to make the color of a dot (or pixel) on the monitor screen.

Each analog color input can be set to one of four (eight) levels by two (three) digital outputs using a simple two-bit (three-bit) digital-to-analog converter (DAC) as shown in Figure 1. The four (eight) possible levels on each analog input are combined by the monitor to create a pixel with one of 4 x 4 x 4 = 64 (8 x 8 x 8 = 512) different colors. So the six (nine) digital control lines let us select from a palette of 64 (256) colors.

VGA Signal Timing

An image (or frame) on a monitor screen is composed of h lines each containing w pixels. VGA frame size is expressed as w x h with typical sizes of 640 x 480, 800 x 600, 1024 x 768 and 1280 x 1024.

In order to paint a frame, there are deflection circuits in the monitor that move the beams of electrons from left-to-right and top-to-bottom across the screen. These circuits require two sync signals in order to start and stop the beams at the right times so that a line of pixels is painted across the monitor and the lines stack up from the top to the bottom to form an image. The timing for the VGA sync signals is shown in Figure 2.

Negative pulses on the horizontal sync signal mark the start and end of a line and ensure that the monitor displays the pixels between the left and right edges of the visible screen area. The pixels are sent on the red, green and blue signal lines within a 25.17 µs window. After this, a *front porch* interval of 0.94 µs is inserted before the horizontal sync signal pulses low for 3.77 µs. After a *back porch* interval of 1.89 µs, the next line of pixels begins. Therefore, a single line of pixels occupies 25.17 µs of a 31.77 µs interval. The red, green and blue signals are blanked during the 6.6 µs interval comprised of the front porch, sync pulse and back porch.

In an analogous fashion, negative pulses on the vertical sync signal mark the start and end of a frame made up of video lines and ensure that the monitor displays the lines between the top and bottom edges of the visible monitor screen. The lines are displayed within a 15.25 ms window. After this, a front porch interval of 0.45 ms is inserted before the vertical sync signal pulses low for 64 µs. After a back porch interval of 1.02 ms, the next frame begins. Therefore, a single frame of pixels occupies 15.25 ms of a 16.784 ms interval. The red, green and blue signals are blanked during the 1.534 ms interval

comprised of the front porch, sync pulse and back porch.

VGA Generator Operation

A high-level block diagram of the VGA generator circuit is shown in Figure 3. An external system writes pixel values into a pixel buffer (a 256 x 16 FIFO in this case). The pixels are extracted from the buffer into a pixel register. A pixel may be 1, 2, 4, 8 or 16 bits wide so the contents of the pixel register are shifted each clock cycle to place the current pixel in the least-significant bit positions. These bits are sent to a colormap circuit that translates the pixel into red, green and blue values which are sent to the external video DAC.

Two pulse generation circuits are used to create the horizontal and vertical sync signals. These circuits are identical save for the parameters that determine the pulse timing. The horizontal sync generator outputs a single-cycle gate signal coincident with the leading edge of the horizontal sync pulse. This gate signal connects to the clock-enable of the vertical sync generator so it only updates its timing counter once per line of pixels. The gate signal of the vertical sync generator is used as an end-of-frame indicator to the external source of pixel data. It also resets the pixel buffer and clears its contents so the VGA generator starts from a completely cleared state on every frame.

The sync generators also output the horizontal and vertical blanking signals. These are logically-OR'ed to create a global blanking signal. The blanking signals are also combined with the lower order bits of the horizontal pixel counter to determine when to read more pixels from the buffer. For example, if the pixels are four bits wide, then a 16-bit word is needed from the buffer once every four clock cycles. So, the read operation is initiated whenever the video signal is not being blanked and the lower two bits of the pixel counter are both zero.

A full signal is sent to the external pixel data source to let it know when to stop filling the buffer. For a 255-entry FIFO, the full signal is raised when the upper five bits of the FIFO level signal are equal to 11111. This leaves seven empty slots in the FIFO to serve as a safety buffer for pixel data that the external source may already be generating in its own pipeline.

An example of the sequence of operations performed by the VGA generator with four-bit wide pixels is shown in Figure 4. At cycle N in the active interval of a line of video, the counter in the horizontal sync generator will output a value of N. If the least-significant two bits of the counter are both zero, then the rd input to the pixel buffer will be driven high to initiate the reading of four more pixels, $p_N \ldots p_{N+3}$. The buffer read operation begins at the rising edge of clock cycle $N+1$ and the 16-bits of pixel data are available on the buffer outputs at the rising edge of clock $N+2$. At this point, the new pixel data is loaded into the pixel register and the least-significant four bits containing pixel p_N are sent to the colormap circuitry. The colormap circuit computes the RGB values, which are then latched at the beginning of cycle $N+3$. Simultaneously, pixel p_{N+1} is shifted into the lower four bits of the pixel register and the colormap begins to compute the RGB values that will be output during cycle $N+4$.

The colormap circuit takes the bits from each pixel and generates the RGB component values as shown in Table 1. An explanation of the RGB calculation for each possible pixel width follows:

16-bit pixels: For the case where each color component is driven by a three-bit DAC, the nine bits of color information are packed into the lower portion of the pixel while the upper seven bits are ignored. For the two-bit DAC, the six bits of color information are distributed such that the most-significant bit of each component is in the same bit position as the most-significant bit for the three-bit DAC, while the bits at locations 0, 3 and 6 are ignored. The result is that the two-bit DAC will span the same intensity range for each color component as the three-bit DAC, but with only half the gradations within that range. This allows the same image file to be used for any XSA Board regardless of the DAC width.

8-bit pixels: The 16-bit pixel register is divided into two 8-bit pixels. In the case of the three-bit DAC, the nine bits of color information won't fit within the pixel so the least-significant bit of the green component is omitted and the colormap circuit forces DAC bit G0 to zero. As in the 16-bit pixel case, the six bits of color information for the two-bit DAC are distributed such that the most-significant bit of each component is in the same bit position as the most-significant bit for the three-bit DAC, while the bits at locations 0 and 5 are ignored.

4-bit pixels: The 16-bit pixel register is divided into four 4-bit pixels. For both two-bit and three-bit DACs, a single bit in each pixel drives all three bits of each color component. Thus, each color component is either off or driven at full intensity. This means that a four-bit pixel can only display eight possible colors:

black (000), blue (001), green (010), cyan (011), red (100), magenta (101), yellow (110) or white (111).

2-bit pixels: The 16-bit pixel register is divided into eight 2-bit pixels. For the two-bit DAC, the upper bit of the pixel becomes the most-significant bit of all the color components while the lower bit maps to the least-significant bit of all the components. This allows each two-bit pixel to display four levels of gray: 100% (black), 66%, 33% and 0% (white). The case of the three-bit DAC is handled similarly except the least-significant bit of each color component is forced to zero. Thus, the three-bit DAC will display gray-levels of 100% (black), 29%, 57%, and 86% (off-white).

1-bit pixels: The 16-bit pixel register is divided into sixteen 1-bit pixels. For both two-bit and three-bit DACs, the single bit in the pixel drives all the bits in each color component. Thus each pixel displays either black (0) or white (1).

VGA Generator Parameters and I/O

Generic Parameters

The following generic parameters affect the operation of the VGA generator:

FREQ: This parameter sets the master operating frequency of the controller (in units of KHz). This frequency is used to calculate the time delays for the horizontal and vertical sync pulses. The default value for the XSA-50 and XSA-100 Boards is 50,000 KHz and for the XSA-200 and XSA-3S1000 it is 100,000 KHz.

CLK_DIV: Setting this parameter to a value greater than 1 reduces the rate at which pixels are fetched from the buffer and delivered to the monitor. For example, if FREQ=50_000 and CLK_DIV=2, then pixels are sent to the monitor at a rate of 25,000,000 per second.

PIXEL_WIDTH: This parameter gives the bit-width of a pixel. Allowable values are 1, 2, 4, 8 and 16.

PIXELS_PER_LINE: This parameter determines the number of pixels displayed in the active portion of each video scanline.

LINES_PER_FRAME: This parameter sets the number of lines of pixels displayed in the active portion of each video frame.

NUM_RGB_BITS: This is the number of bits of color information in each of the red, green and blue components. Set this to 2 for the XSA-50 and XSA-100 Boards, and 3 for the XSA-200 and XSA-3S1000.

FIT_TO_SCREEN: If this Boolean parameter is set to true, then the sync generators are configured so that the given number of pixels and lines fill the entire width and height of the screen, respectively. Setting this parameter to false, however, embeds the active video region into a standard 31 KHz / 60 Hz horizontal / vertical video frame.

I/O Ports

The host- and monitor-side connections for the VGA generator are shown in Figure 3. The functions of the I/O signals are as follows:

rst: This active-high, asynchronous input resets the internal circuitry of the sync generators.

clk: This is the main clock input. The clock from the external oscillator enters the FPGA through a global clock input pin and drives this input.

pixel_data_in: 16-bit data containing one or more pixels enters the pixel buffer through this bus.

eof: This active-high output indicates when the display of a video frame has been completed and pixels for the next frame can begin entering the buffer.

full: This active-high output indicates when the pixel buffer is full and no more space is currently available for more pixels.

vsync_n: This active-low output drives the vertical sync input of a VGA monitor.

hsync_n: This active-low output drives the horizontal sync input of a VGA monitor.

blank: This active-high output signals when the red, green and blue video signals are blanked.

r: This bus carries the data bits for the red video component to a DAC whose analog output is delivered to the VGA monitor.

g: This bus carries the data bits for the green video component to a DAC whose analog output is delivered to the VGA monitor.

b: This bus carries the data bits for the blue video component to a DAC whose analog output is delivered to the VGA monitor.

VGA Generator Test Application

A simple application of the VGA generator combines it with an SDRAM controller to display images stored in the SDRAM on the XSA Board.

Figure 5 shows the connections between the VGA generator and the SDRAM controller. The counter stores the next SDRAM address that will be read when the pixel buffer is not full. Once the read operation starts, the earlyOpBegun signal goes high and the address counter increments to point to the next word of image data while the current read operation proceeds. Once the data is available from the SDRAM, the rdDone signal triggers the write-enable that loads the data into the pixel buffer. This sequence of operations is repeated for the image data at the next SDRAM address. Once a complete video frame has been displayed, the eof signal goes high and resets the counter to the start of the image data and the entire process repeats.

The source files for this application can be found at http://www.xess.com/projects/an-101204-vgagen.zip.

Help With Video Problems

You might set the generic parameters and then find that your monitor will not sync with the output of the VGA generator or the image is distorted. Here are some solutions for common problems.

Monitor Won't Sync

This should only happen if you set FIT_TO_SCREEN to true. (When FIT_TO_SCREEN is false, the VGA generator outputs standard 31 KHz / 60 Hz horizontal / vertical sync signals that any monitor should accept.) In this mode, the period of the horizontal scan line is set to

(#pixels x CLK_DIV) / pixel clock freq. + 6 μs

where the additional 6 μs consists of 1 μs for the front porch, 4 μs for the horizontal sync pulse and 1 μs for the back porch. If your number of pixels per line is too large or your pixel clock is too low (or your

CLK_DIV value is too large), then the period will be too long and the monitor won't sync. You can also have too few pixels per line such that the period is too short, but monitors tend to be more tolerant of shorter periods (higher horizontal frequencies) than longer periods.

In the same vein, you can specify too many scan lines per frame and get a frame rate that is too small. The period for a video frame is:

(#lines x CLK_DIV) / horizontal freq. + 1424 μs

where the additional 1424 μs consists of 340 μs for the front porch, 64 μs for the horizontal sync pulse and 1020 μs for the back porch. Monitors tend to be more tolerant of shorter vertical periods (higher frame rates) than longer periods.

A spreadsheet is included in the VGA generator source files that calculates the horizontal and vertical frequencies based on the pixel clock frequency and the screen width and height. You can use it to find values that will bring the sync frequencies into the range that is acceptable to your monitor.

Non-Black Background or Strange Pixel Colors

This usually happens if FIT_TO_SCREEN is false and you are trying to fit too many pixels into a scanline. This causes the active video region to extend into the horizontal sync pulse interval. Most monitors sample the video signal during the horizontal and vertical sync pulses to get the baseline blanking level (the video should be blanked during these pulses). The monitor uses the blanking level as a baseline to compare to the video signal level during the active video intervals. If the baseline is not at the blanking level, then the monitor will not have an accurate reference against which to judge the levels of the active video RGB components and the result will be pixels displayed in off-colors against a non-black background.

If the horizontal sync front porch interval is negative when you enter your screen dimensions and pixel clock frequency into the vga-timing.xls spreadsheet, then you may experience this problem. You can fix it by using fewer pixels per line or a faster clock.

Flattened or Elongated Graphics

This will occur if your ratio of pixels per line to lines per frame deviates from 4/3. If you have too few lines such that the ratio exceeds 4/3, then the height

of each line must be stretched to cover the screen, resulting in non-square pixels that are taller than they are wide. Conversely, having too many scan lines squashes the pixels so they are shorter than they are tall. In general, strive to maintain the 4/3 pixel/line ratio.

Unstable or Noisy/Streaky Image

This can occur if the pixel buffer runs out of pixels during the active portion of a video frame. Then the buffer will repeatedly deliver the last pixel that was output and this shows up as a horizontal line of constant color. The line will end once new data enters the buffer, but the rest of the frame will be offset horizontally. Since the buffer usually doesn't become empty at the exact same point in each frame, successive frames will be affected at different points and this will make the video appear to "jump".

The solution is to prevent the pixel buffer from emptying. This can be done in two ways:

1. Run the VGA generator at a slower clock rate by setting CLK_DIV to 2 or more. This will decrease the rate at which pixels leave the buffer so it will be easier for the external system (which is still running at the higher clock frequency) to keep up with the demand for pixels. This solution will only work if the slower rate at which pixels are read from the buffer is still sufficient to meet the timing and display resolution requirements of the monitor.

2. Pack more pixels into each pixel buffer entry. This will decrease the rate at which the buffer empties because each buffer entry contains multiple pixels. This solution will only work if the smaller pixels still have sufficient color resolution for the application.
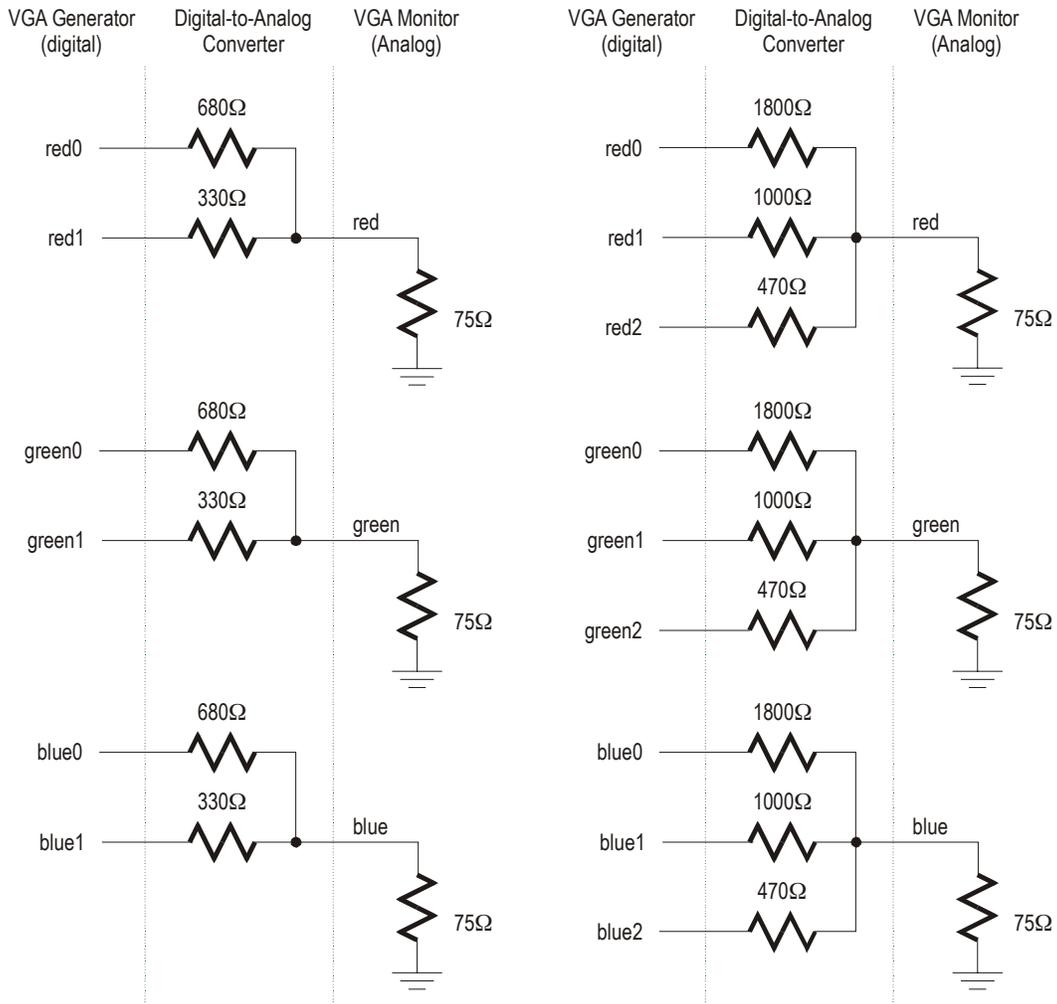
Figure 1: Two-bit and three-bit digital-to-analog VGA monitor interfaces.
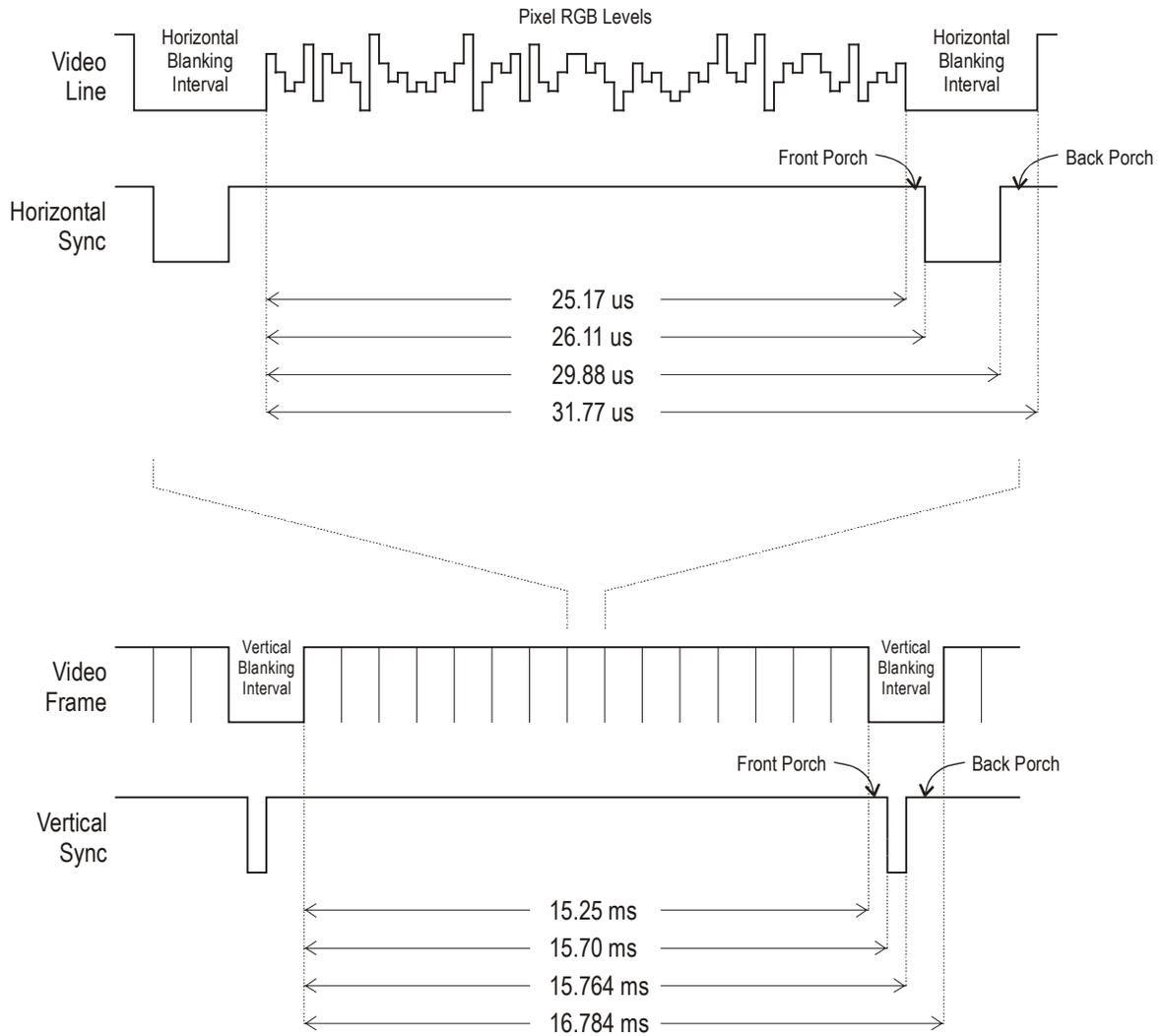
Figure 2: VGA signal timing.
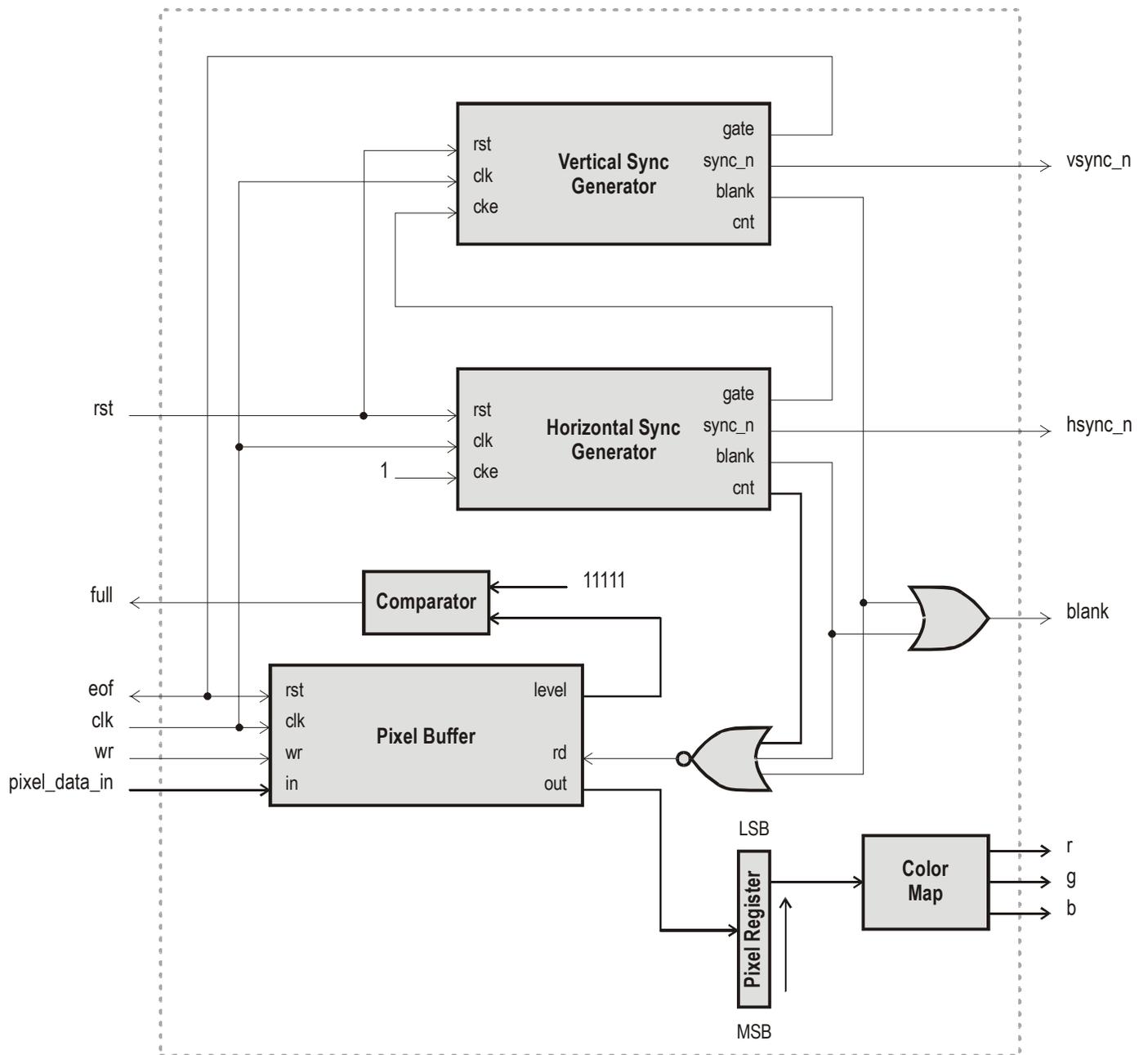
Figure 3: VGA generator block diagram.

N   N+1   N+2   N+3

pixel clk

horiz. cnt   N   N+1   N+2   N+3

pixel buffer rd

pixel buffer out $p_{N+3},p_{N+2},p_{N+1},p_N$

pixel register $0,0,p_{N-1},p_{N-2}$ $0,0,0,p_{N-1}$ $p_{N+3},p_{N+2},p_{N+1},p_N$ $0,p_{N+3},p_{N+2},p_{N+1}$

R,G,B $R_{N-3}, G_{N-3}, B_{N-3}$ $R_{N-2}, G_{N-2}, B_{N-2}$ $R_{N-1}, G_{N-1}, B_{N-1}$ R_N, G_N, B_N

initiate read of pixel buffer   *latch buffer output into pixel register*   *latch colormap to RGB outputs*
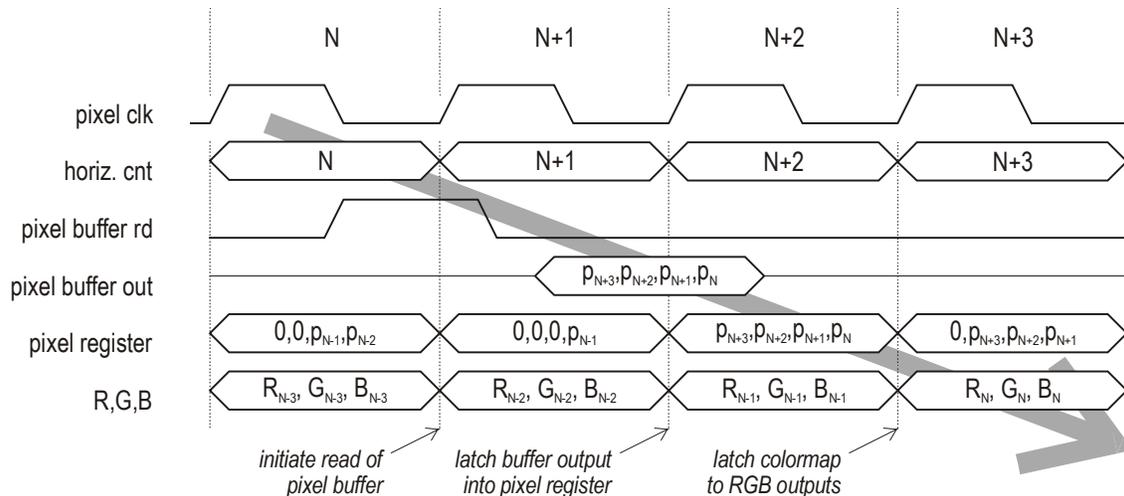
Figure 4: Pipelined operations in the VGA generator.

Table 1: Assignment of bits in the pixel register to the inputs of the RGB DACs for various pixel widths.

#Pixel Bits / #RGB Bits	Pixel N															
16/2						R1	R0		G1	G0		B1	B0		
16/3						R2	R1	R0	G2	G1	G0	B2	B1	B0	
	Pixel N+1								Pixel N							
8/2	R1	R0		G1	G0	B1	B0		R1	R0		G1	G0	B1	B0	
8/3	R2	R1	R0	G2	G1	B2	B1	B0	R2	R1	R0	G2	G1	B2	B1	B0
	Pixel N+3				Pixel N+2				Pixel N+1				Pixel N			
4/2		R1,R0	G1,G0	B1,B0		R1,R0	G1,G0	B1,B0		R1,R0	G1,G0	B1,B0		R1,R0	G1,G0	B1,B0
4/3		R2,R1,R0	G2,G1,G0	B2,B1,B0		R2,R1,R0	G2,G1,G0	B2,B1,B0		R2,R1,R0	G2,G1,G0	B2,B1,B0		R2,R1,R0	G2,G1,G0	B2,B1,B0
	Pixel N+7		Pixel N+6		Pixel N+5		Pixel N+4		Pixel N+3		Pixel N+2		Pixel N+1		Pixel N	
2/2	R1,G1,B1	R0,G0,B0	R1,G1,B1	R0,G0,B0	R1,G1,B1	R0,G0,B0	R1,G1,B1	R0,G0,B0	R1,G1,B1	R0,G0,B0	R1,G1,B1	R0,G0,B0	R1,G1,B1	R0,G0,B0	R1,G1,B1	R0,G0,B0
2/3	R2,G2,B2	R1,G1,B1	R2,G2,B2	R1,G1,B1	R2,G2,B2	R1,G1,B1	R2,G2,B2	R1,G1,B1	R2,G2,B2	R1,G1,B1	R2,G2,B2	R1,G1,B1	R2,G2,B2	R1,G1,B1	R2,G2,B2	R1,G1,B1
	N+15	N+14	N+13	N+12	N+11	N+10	N+9	N+8	N+7	N+6	N+5	N+4	N+3	N+2	N+1	Pixel N
1/2	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B
1/3	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B	R,G,B
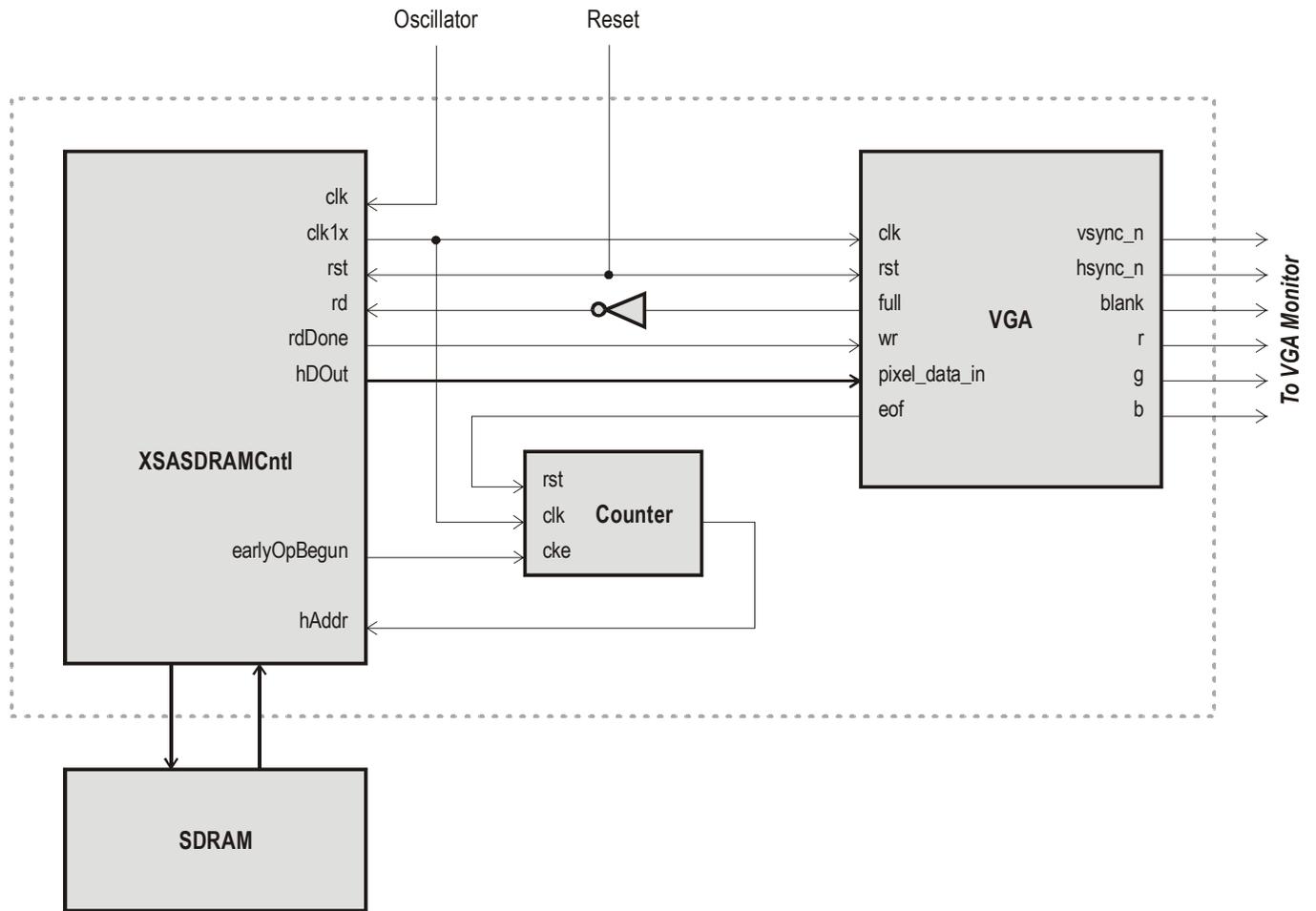
Figure 5: Using the VGA generator to display images stored in SDRAM.

Revision	Date	Comments
1.0	10/12/04	Initial version.