Build a VGA Monitor Controller

Building a video controller circuit is simpler than you think. Enoch built a VGA monitor controller with just two 10-bit binary up counters, four SR flip-flops, and 11 AND gates. Although it isn't a replacement for a graphics card, it's a great way to display images.

Understanding video signals and building video controller circuits is always a challenge. But things are getting easier. I recently took another look at the VGA video signal and realized that I could build a VGA monitor controller with just two binary counters, four flip-flops, and 11 AND gates. Yes, that's right, just two 10-bit binary up counters, four SR flip-flops, and 11 AND gates! Now, of course, this isn't a replacement for your high-end graphics card in your PC, nor is it even a low-end video card, but it's capable of displaying images on a standard VGA monitor. Most importantly, this simple VGA monitor controller circuit allows you to easily understand how the VGA monitor works and how to control it. When I introduced this to my introductory digital logic design class, the students were totally amazed that it could be this simple.

PIXELS ON SCREEN

To begin, you need to understand how a VGA monitor works. The monitor screen for a standard VGA format contains 640 columns by 480 rows of picture elements called pixels (see Figure 1). An image is displayed on the screen by turning on and off individual pixels. Turning on one pixel doesn't represent much, but combining numerous pixels generates an image. The monitor continuously scans through the entire screen, rapidly turning individual pixels on and off. Although pixels are turned on one at a time, you get the impression that all the pixels are on because the monitor scans so

quickly. This is why old monitors with slow scan rates flicker.

Figure 1 shows that the scanning starts from row 0, column 0 in the top left corner of the screen and moves to the right until it reaches the last column. When the scan reaches the end of a row, it retraces to the beginning of the next row. When it reaches the last pixel in the bottom right corner of the screen, it retraces back to the top-left corner and repeats the scanning process. In order to reduce flicker on the screen, the entire screen must be scanned 60 times per second (or more). During the horizontal and the vertical retraces, all the pixels are turned off.

FIVE CONROL SIGNALS

The VGA monitor is controlled by five signals: red, green, blue, horizontal synchronization, and vertical synchronization. The three color signals, collectively referred to as the RGB signal, control the color of

a pixel at a given location on the screen. They are analog signals with voltages ranging from 0.7 to 1 V.

Different color intensities are obtained by varying the voltage. For simplicity, your circuit will treat these three-color signals as digital signals, so you can just turn each one on or off. As a result, the circuit is capable of displaying only eight colors (2^3 = 8). The digital-to-analog conversion circuit is shown in Figure 2a. Figure 2b shows a slightly enhanced digital-to-analog converter circuit that can display up to 64 colors (2^6).

The horizontal and vertical synchronization signals are used to control the timing of the scan rate. Unlike the three analog RGB signals, these two sync signals are digital signals. In other words, they take on either a logic 0 or a logic 1 value. The horizontal synchronization signal determines the time it takes to scan a row, while the vertical synchronization signal determines the time it takes to scan the entire screen. Understanding how to control a VGA monitor simply boils down to understanding the timings for these two synchronization signals. By manipulating these two sync signals and the three RGB signals, images are formed on the monitor screen.

SYNC TIMINGS

The horizontal and vertical synchronization signal-timing diagram is shown in Figure 3. When inactive, both
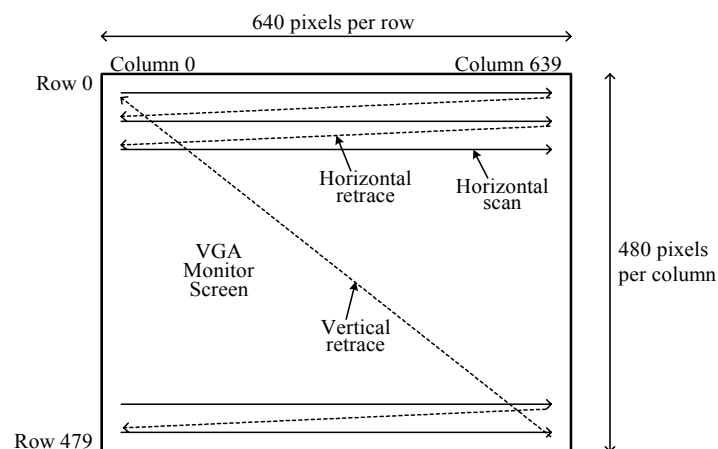

Figure 1—The VGA monitor has 640 columns and 480 rows. Scanning starts from row 0, column 0 and moves to the right and down until reaching row 479, column 639
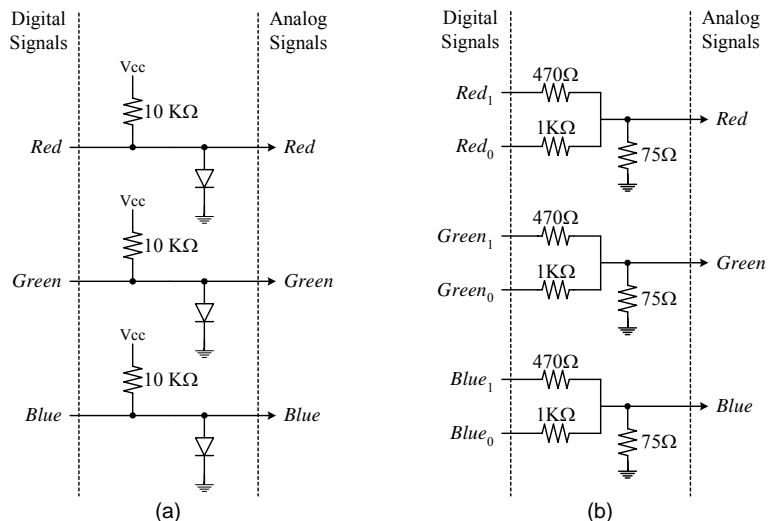
Figure 2—D/A converter circuits drive the RGB signals. You can use them to produce eight colors **(a)** or 64 colors **(b)**. The VGA monitor controller controls the digital signals. The analog signals are connected to the VGA monitor via a 15-pin D-Sub connector.

synchronization signals are at a logic 1 value. A row scan begins with the horizontal sync signal going low for 3.77 μs (see section B in Figure 3). A 1.79-μs high on the signal follows this (see section C in Figure 3). Next, the data for the three color signals is sent, one pixel at a time, for the 640 columns for 25.42 μs. Finally, after the last column pixel, there is another 0.79 μs of inactivity on the RGB signal lines for the horizontal retrace before the horizontal sync signal goes low again for the next row scan. The total time to complete one row scan is 31.77 μs.

The timing for the vertical sync signal is analogous to the horizontal

one. The 64-μs active low vertical sync signal resets the scan to the top-left corner of the screen (see section P in Figure 3). A 1,020-μs high follows this on the signal. Next, there are the 480 31.77-μs row scans, giving a total of 15,250 μs (480 × 31.77), as shown in section R.

Finally, after the last row scan, there are another 450 μs before the vertical sync signal goes low again to start another complete screen scan in the top left corner. It takes a total of 16,784 μs to complete one full screen scan.

To get the monitor operating properly, simply get the horizontal and vertical sync signals timing correct

and then send out the RGB data for each pixel at the right column and row position. For example, if you want to turn on the red pixel at row 13 and column 48, wait for the scan to reach row 13 and column 48 and then set the red signal to logic 1. To accomplish this, you need to generate the horizontal and vertical sync signals correctly based on the timing diagrams shown in Figure 3. You also must keep track of the current row and column counts so that you know where the scan is. It turns out that you can do both of these things using the same component, which is the binary up counter. You need two counters. One is for generating the horizontal sync and keeping tract of the column count. The second is for generating the vertical sync and keeping track of the row count.

COUNTING CLOCK CYCLES

Getting the correct timing for the two synchronization signals is simple if you use the correct clock frequency. To obtain the 480 × 640 screen resolution, use a clock with a 25.175-MHz frequency. A higher clock frequency is needed for a higher screen resolution. For the 25.175-MHz clock, the period is the following:

$$\frac{1}{25.175 \times 10^6}$$

or approximately 0.0397 μs per clock cycle. For section B of the horizontal synchronization signal, you need 3.77 μs, which is approximately 95 clock cycles (3.77/0.0397). For section C in Figure 3, you need 1.79 μs, which is approximately 45 clock cycles. Similarly, you need 640 clock cycles (section D) for the 640 columns of pixels and 20 clock cycles for section E.

The total number of clock cycles needed for each row scan is 800 clock cycles (95 + 45 + 640 + 20). Notice that with a 25.175- MHz clock, section D requires exactly 640 cycles, generating the 640 columns per row. If you use a different clock speed, you will get a different screen resolution. The number of clock cycles required by the four regions in the horizontal sync signal is summarized in Table 1.
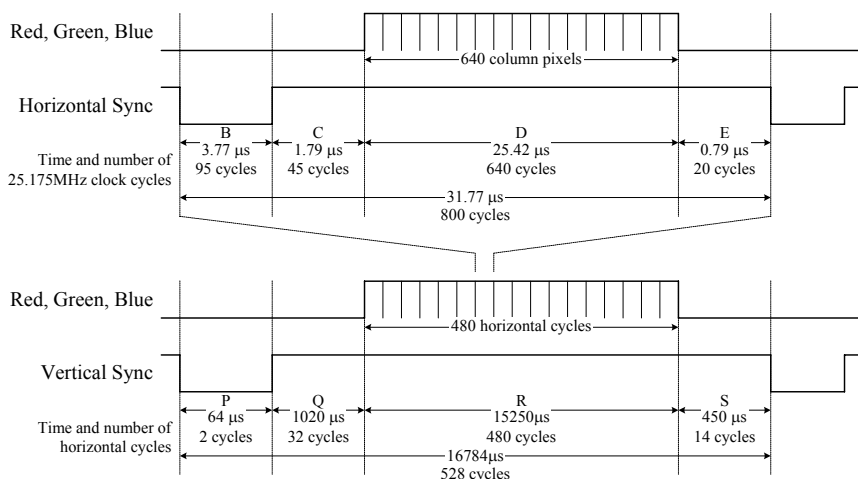

Figure 3—The horizontal and vertical synchronization signal timing diagram uses a 25.175-MHz clock. Each of the two signals has four regions: B, C, D, and E for the horizontal sync, and P, Q, R, and S for the vertical sync. Controlling the VGA monitor involves getting the correct timing for the regions.

Because the vertical sync signal is analogous to the horizontal sync signal, you can perform the same calculations as with the horizontal sync regions to obtain the number of cycles needed for each vertical region. However, instead of using the number of periods of a 25.175-MHz clock, the times for each vertical region are multiples of the horizontal cycle. For example, the time for a horizontal cycle is 31.77 µs, and section P requires 64 µs, which is approximately two horizontal cycles (2 × 31.77). Section Q requires 1,020 µs, which equals 32 horizontal cycles (1,020/31.77). The calculation for section R is 480 horizontal cycles (15,250 µs/31.77 µs). Of course, it has to be exactly 480 times, because you need to have 480 rows per screen. The number of horizontal cycles required by the four regions in the vertical sync signal is also summarized in Table 1.

If you use a 25.175-MHz clock to drive a counter so that it increments at every clock cycle, all you have to do to get the correct horizontal sync signal is count the correct number of cycles for each region. Starting the count at zero, set the horizontal sync signal (H_Sync_out) to zero (for low). When the count reaches 95, set H_Sync_out to one (for high). When the count reaches 140 (95 + 45), keep H_Sync_out at one. When the count reaches 780 (95 + 45 + 640), continue to keep H_Sync_out at one. Finally, when the count reaches 800 (95 + 45 + 640 + 20), set H_Sync_out to zero, and reset the counter to zero. This completes one period of the H_Sync_out signal.

Similarly, you can use another counter for the vertical sync signal. The clock for this counter is derived from the horizontal counter so that the vertical counter counts once for each horizontal cycle.
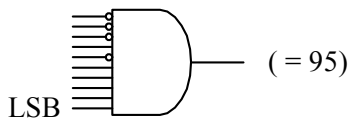

Figure 4—A 10-input AND gate is connected to test whether or not a number is equal to 95 (0001011111 in binary). If the input is 95, the AND gate outputs a one; otherwise, it outputs a zero.

	B	C	D	E	Total
Time	3.77 µs	1.79 µs	25.42 µs	0.79 µs	31.77 µs
Number of a 25.175MHz clock cycles	95 cycles	45 cycles	640 cycles	20 cycles	800 cycles

	P	Q	R	S	Total
Time	64 µs	1020 µs	15250 µs	450 µs	16784 µs
Number of horizontal cycles	2 cycles	32 cycles	480 cycles	14 cycles	528 cycles

Table 1—Take a look at the number of cycles needed for the different regions of the horizontal and vertical sync signals.

VGA CONTROLLER CIRCUIT

Two 10-bit binary up counters are needed for the horizontal and vertical sync signals. A 9-bit counter can only count up to 512 (2^9), but you need to count up to 528 and 800 for the vertical and horizontal sync signals respectively. A 10-bit counter can count up to 1,024 (2^{10}). A 10-input AND gate is used for comparing the count with a constant. Figure 4 shows the connection of a 10-input AND gate for comparing with the constant 95. Because 95 equals 0001011111 in binary, bits 6, 8, 9, and 10 (starting from the LSB) of the 10-input AND gate are inverted. Four such comparators are used for the four horizontal regions, each connected according to the ending count value that is to be tested.

Within each region, you need to maintain the value of the horizontal sync signal. For example, at count zero, set H_Sync_out to zero; but between counts zero and 95, H_Sync_out must be kept at zero. An SR flip-flop is used to keep the signal steady. Recall that the SR flip-flop sets the output Q to a one when the input set is asserted with a one. It resets the output Q to a zero when the input reset is asserted with a one. If both set and reset inputs are deasserted with a zero, then the output Q will maintain its current value. Hence, to obtain the horizontal sync signal, you can assert the reset input when the count is zero (or 800) and assert the set input when the count is 95. The Q output of the SR flip-flop is now the H_Sync_out signal.

You can do one of three things to keep track of the column count from zero to 639 in the D region. The first solution is to use another counter that counts from zero to 639 using the

same clock frequency as the horizontal sync counter, but this counter counts only when the horizontal sync counter is in region D. This solution requires an extra counter.

The second solution is to subtract the offset for the B and C regions, so that when the horizontal sync counter reaches 140 (95 + 45), you will subtract 140 to get a zero. Then, 141 minus 140 will produce a one, 142 minus 140 will produce a two, and so on. This solution requires an extra subtraction unit.

The last solution is the best. Simply offset the horizontal sync counter so that you start the count at the beginning of region D instead of starting it at the beginning of region B. At the beginning of region D, the count is reset to zero. At the end of region D, the count will be at 639. This way, when the counter is counting in region D, the count will also represent the correct column count. Hence, the counter will reach 800 at the end of region C.

Putting everything together, you get the circuits shown in Figure 5. Figure 5a shows the horizontal counter with the four AND gates for testing for the four horizontal region values D, D + E, D + E + B, and D + E + B + C. The output of the counter is the column count.

The circuit also outputs a ROLL_OVER signal, which is used to reset the horizontal counter to zero and is also the clock signal for the vertical counter. This signal is asserted each time the counter reaches 800. When the signal is a one, it asserts the counter's load input. When this happens, the 10-bit counter input value D9–0 (which is a constant 0) is loaded into the counter. The count input of the up counter is always set to a one, because you want the counter to
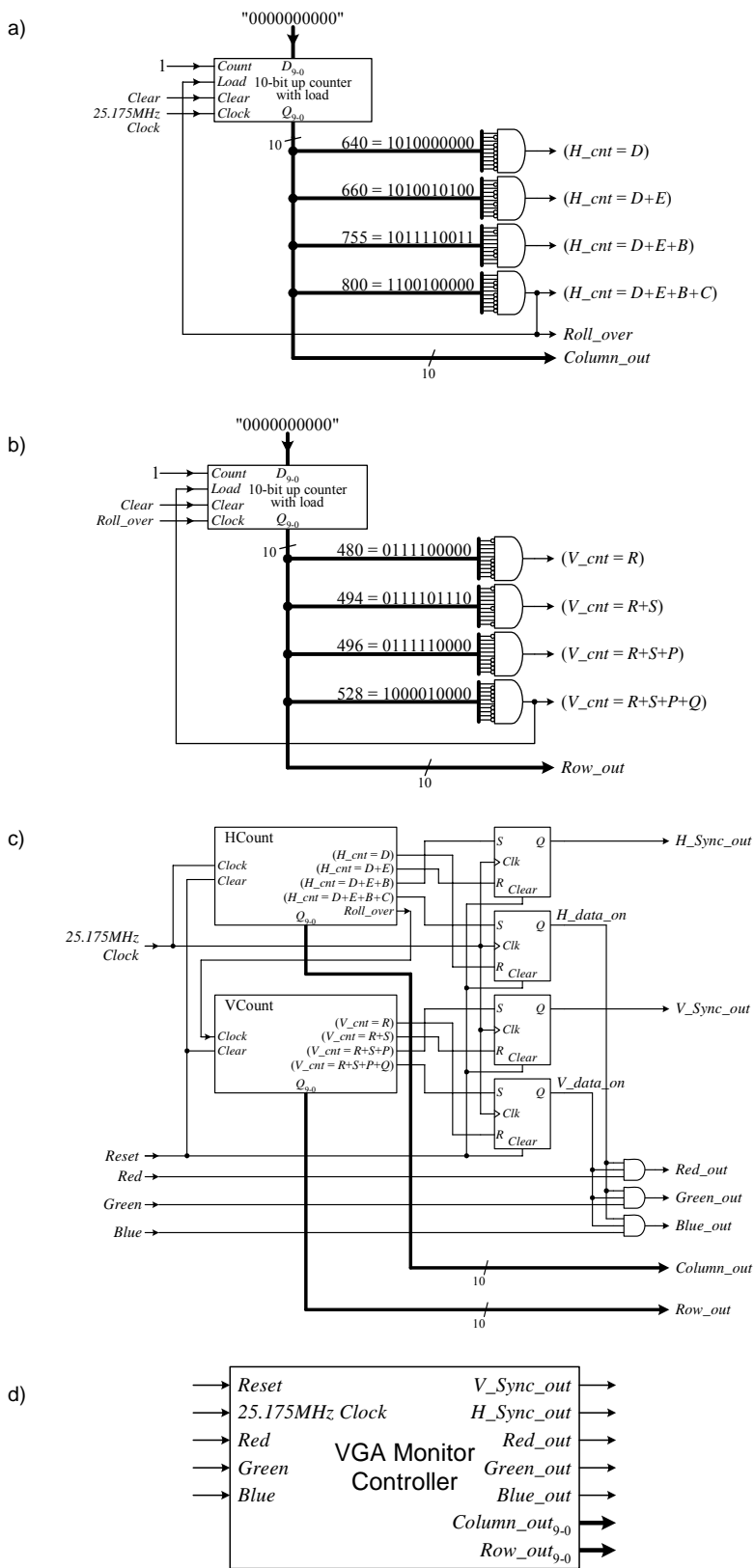
a)

b)

c)

d)

Figure 5—The VGA monitor controller circuit contains the following: a horizontal counter circuit for generating the horizontal sync and column count signals **(a)**; a vertical counter circuit for generating the vertical sync and row count signals **(b)**; a complete VGA controller circuit **(c)**; and a logic symbol for the controller circuit **(d)**.

clear input is connected to a Reset switch. The clock input is connected to a 25.175-MHz clock.

Figure 5b shows the vertical counter. It is almost identical to the horizontal counter circuit, except for the clock and the values tested for by the four AND gates. The clock for this counter is the ROLL_OVER signal from the horizontal counter. The values tested for by the AND gates are the vertical region values R, R + S, R + S + P, and R + S + P + Q.

The complete VGA monitor controller circuit is shown in Figure 5c. The H_DATA_ON and V_DATA_ON signals are generated in a similar fashion to the H_SYNC_OUT and V_SYNC_OUT signals, except they are set to a one when the counters are in the D and R regions. Outside these regions, they are set to a zero.

The H_DATA_ON signal is set to a one when the horizontal counter is at zero (800). It's reset to a zero when the counter is at 640. The V_DATA_ON signal is set to a one when the vertical counter is at zero (528). It's reset to a zero when the counter is at 480. These two DATA_ON signals are used to enable the output of the RBG signals. The RGB signals connected to the monitor must be turned on only when the two sync signals are in regions D and R. Three AND gates, one for each of the three color signals, are used to enable the color signals. The H_DATA_ON and V_DATA_ON signals are the enabler lines to the AND gates.

The logic symbol for the VGA controller is shown in Figure 5d. The H_SYNC_OUT, V_SYNC_OUT, RED_OUT, GREEN_OUT, and BLUE_OUT signals connect directly to pins 13, 14, 1, 2, and 3 of the VGA connector. You can optionally connect a switch to the Reset input. The clock source is a 25.175- MHz clock. To display something on the screen, you need to check the values of COLUMN_OUT and ROW_OUT, and set the RED, GREEN, and BLUE signals accordingly.

CONTROLLER TEST

To turn on a particular pixel, you need to test the values of the column and row counts from the controller. If
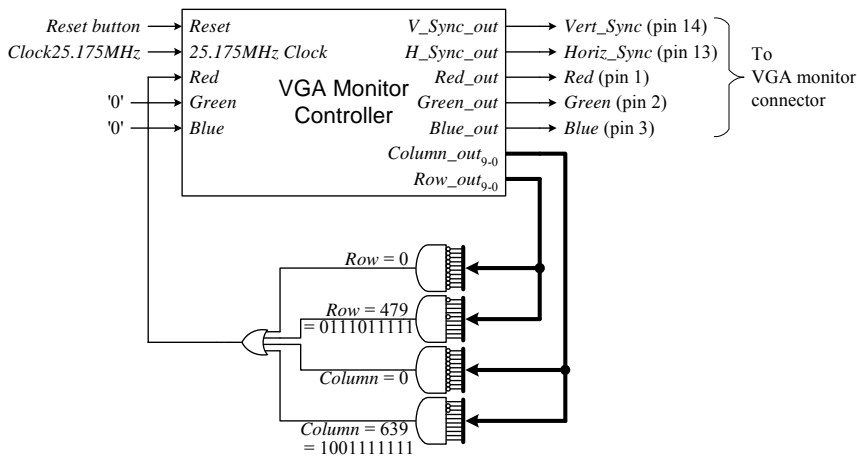
Figure 6—Use this circuit to display a red border around the VGA monitor using the VGA monitor controller.

they are equal to the location of the pixel you want to turn on, then you assert any of the color signals, and that pixel will be turned on with that color. For example, if you want the pixel at column 3, row 5 to be blue, then you need to check the values of COLUMN_ OUT and ROW_OUT from the controller to see if they are equal to three and five respectively. If they are, set the BLUE input signal to a one; otherwise, set it to a zero.

Figure 6 shows the circuit for displaying a red border around the monitor using the VGA monitor controller. Four AND gates are used to test for the column and row border values. Because the screen resolution is 480 × 640, the four border values to test are column = 0, column = 639,

row = 0, and row = 479. If one of these tests is true, then set the red signal to a one.

Instead of using discrete ICs for constructing the controller circuit, I implemented the controller on an FPGA chip using Altera's UP2 development board. The board has a builtin VGA connector with the five signal pins connected to the FPGA chip. The VGA monitor controller, along with a demonstration test circuit for creating a screen image, is implemented in the FPGA chip on the board.

Photo 1 shows the UP2 board having the controller circuit and a demonstration test circuit implemented in the FPGA. The demonstration test circuit generates a red border, two blue letters, and a green square on the monitor screen. Rather than manually connecting the numerous AND and OR gates needed for comparing with the various column and row values to turn on the RGB signals, I have written a VHDL code for the test circuit. The complete test circuit code for generating the image is shown in Listing 1. After synthesizing the code, the resulting netlist, along with the monitor controller circuit, is downloaded to the FPGA chip. The result is shown on the monitor in Photo 1.

In order to display more

complex images, memory is used to keep track of what pixel should be turned on or off and for which color (instead of using numerous AND gates as comparators to check for the current column and row values). If you have one memory location for each color of each pixel, you can use the column and row counts from the controller as the address for the memory. The content of the memory location will be the value for the color signals.

Enoch Hwang has a Ph.D. in computer science. He is currently an associate professor at La Sierra University and a lecturer at the University of California, Riverside. Enoch's interests include embedded microprocessor systems, automation, and robotics. You may reach him at ehwang@lasierra.edu.
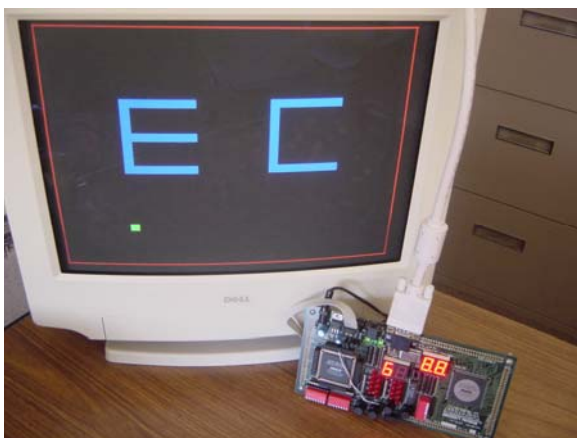

Photo 1—The VGA monitor screen shows a red border, two blue letters, and a green square. The UP2 development board, which contains the FPGA chip with the VGA monitor controller circuit, outputs the video signals to the monitor.

PROJECT FILES
To download the code, to to ftp.circuitcellar.com/pub/Circuit_Cellar/2004/172

SOURCE
UP2 Development kit Altera Corp. www.altera.com
Digital Logic and Microprocessor Design www.lasierra.edu/~ehwang/digitaldesign

```
LIBRARY IEEE;
USE  IEEE.STD_LOGIC_1164.ALL;

ENTITY demo IS PORT (
    Column,Row: IN STD_LOGIC_VECTOR(9 DOWNTO 0);
    Red,Green,Blue: OUT STD_LOGIC);
END demo;

ARCHITECTURE Dataflow OF demo IS

BEGIN
    -- Red boarder
    Red <= '1' WHEN
        (Row(9 DOWNTO 0) = "0000000000") OR            -- rows 0
        (Row(9 DOWNTO 0) = "0111011111") OR            -- rows 479
        (Column(9 DOWNTO 0) = "0000000000") OR  -- columns 0
        (Column(9 DOWNTO 0) = "1001111111")            -- columns 639
    ELSE '0';

    -- Green small square
    Green <= '1' WHEN
        ((Row(9 DOWNTO 4) = "011000") AND         -- use only the first 6 most
        (Column(9 DOWNTO 4) = "001000"))          -- significant bits giving a
    ELSE '0';                                     -- 16 x 16 pixel square

    -- Blue letters "E C"
    Blue <= '1' WHEN                              -- this is just setting
            -- Blue to 1 when any one of the following conditions is true
        ((Row(9 DOWNTO 4) = "001000") AND
        ((Column(9 DOWNTO 7) = "001") OR (Column(9 DOWNTO 7) = "011"))) OR

        ((Row(9 DOWNTO 4) = "001001") AND
        ((Column(9 DOWNTO 4) = "001000") OR (Column(9 DOWNTO 4) = "011000"))) OR

        ((Row(9 DOWNTO 4) = "001010") AND
        ((Column(9 DOWNTO 4) = "001000") OR (Column(9 DOWNTO 4) = "011000"))) OR

        ((Row(9 DOWNTO 4) = "001011") AND
        ((Column(9 DOWNTO 4) = "001000") OR (Column(9 DOWNTO 4) = "011000"))) OR

        ((Row(9 DOWNTO 4) = "001100") AND
        ((Column(9 DOWNTO 7) = "001") OR (Column(9 DOWNTO 4) = "011000"))) OR

        ((Row(9 DOWNTO 4) = "001101") AND
        ((Column(9 DOWNTO 4) = "001000") OR (Column(9 DOWNTO 4) = "011000"))) OR

        ((Row(9 DOWNTO 4) = "001110") AND
        ((Column(9 DOWNTO 4) = "001000") OR (Column(9 DOWNTO 4) = "011000"))) OR

        ((Row(9 DOWNTO 4) = "001111") AND
        ((Column(9 DOWNTO 4) = "001000") OR (Column(9 DOWNTO 4) = "011000"))) OR

        ((Row(9 DOWNTO 4) = "010000") AND
        ((Column(9 DOWNTO 7) = "001") OR (Column(9 DOWNTO 7) = "011")))
    ELSE '0';
END Dataflow;
```

Listing 1—Use this VHDL code to generate a red border, two blue letters ("EC"), and a green square. The code simply tests the values of the row and column counts and sets the three color signals (red, green, and blue) accordingly.