

VGA Graphics Hardware

Department of Electrical and Computer Engineering

University of Dayton

Adam Dulay

ECE 499 Special Topic In Computing

1 Introduction

This is ultimately a diversion from the main scope of the project, but if implemented properly, should make a for interesting demonstrations. During this section, the VHDL and subsequent hardware required for producing a working video graphics array (VGA) signal will be outlined.

Below, in Figure 1, is a diagram that describes the signals and timing requirements for a VGA signal. In order to produce a working VGA signal, the synchronization pulses must first be produced based on the resolution of the desired display. For this first demonstration, an 800x600 pixel resolution was implemented because it natively used a 50MHz pixel clock frequency; which the DE2-115 board (DE2) also natively has onboard. Based on the specification from *tinyvga.com* (see references), this amounted to having the horizontal sync pulse be for 120 pixels, the vertical sync pulse be for 6 pixels, the horizontal front porch be for 56 pixels, the vertical front porch be for 37 pixels, the horizontal back porch be for 64 pixels, and the vertical back porch be for 23 pixels. These synchronization signals were all produced on the FPGA and are directly outputted to the monitor unlike the color signals. The color data bits and an extension to the pixel clock were then fed into the onboard digital to analog converter (DAC) unit along with the blanking signal. The onboard DAC unit is an ADV7123_a, which accepts 8bits of digital color data from the red, green, and blue channels respectively, and converts them into an analog signal that is outputted to the monitor (this is done in the place of the more archaic resistor ladder approach that many other systems use). The DAC unit is fed the clock signal and the “blank” signal (as can be seen in the VHDL module), because all analog color output must cease during the blanking intervals otherwise the monitor will not correctly display the image. Below in Figures 2 and 3 are the functional diagram for the DAC unit as well as the clock diagram of the VGA datapath on the DE2 respectively.

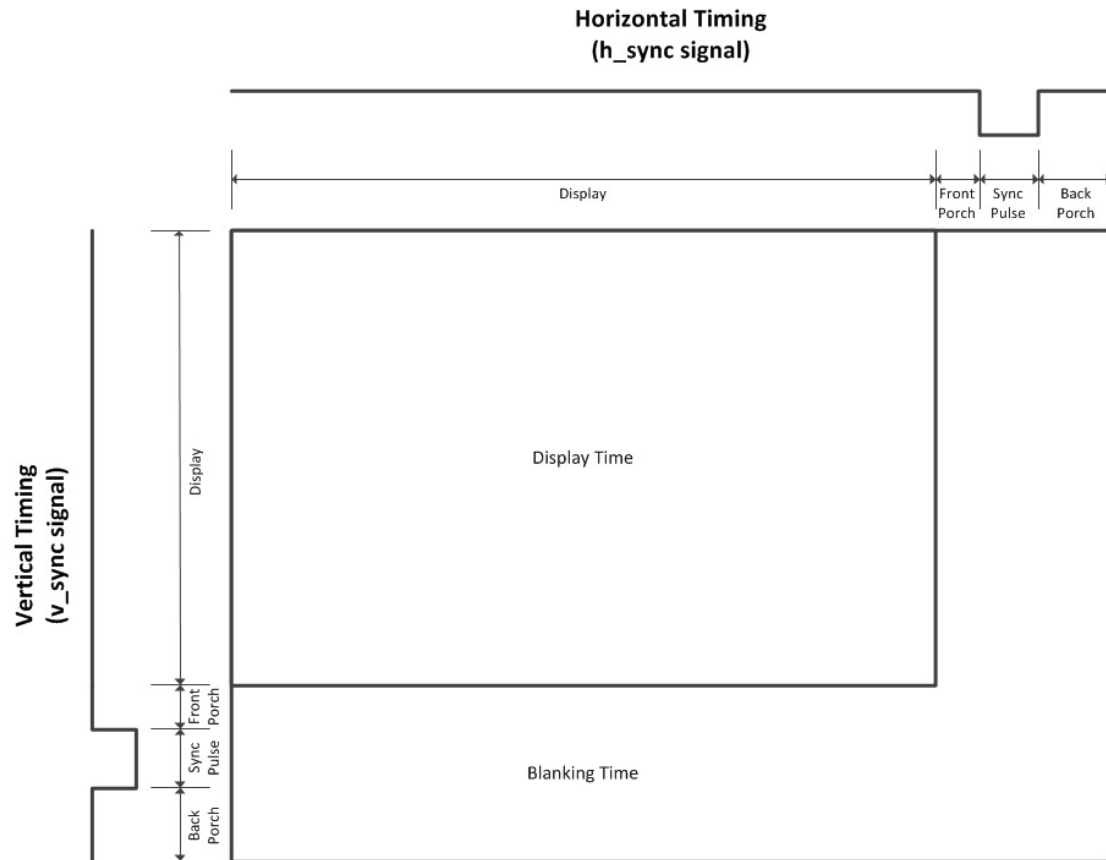


Figure 1. VGA Signal Timing Diagram

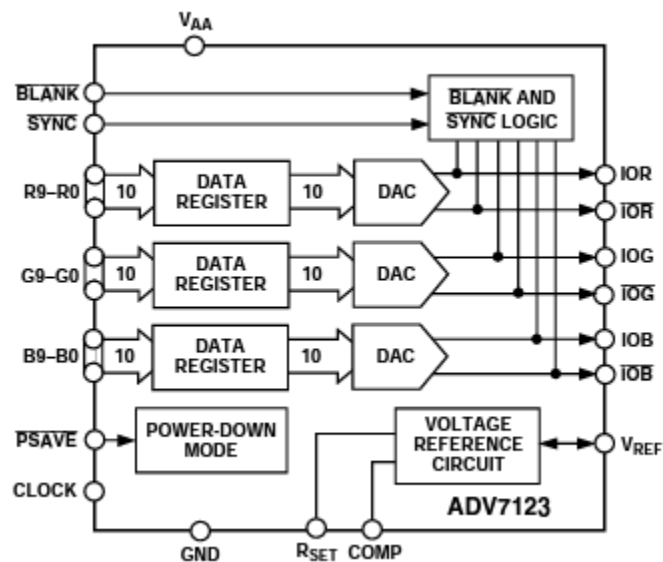


Figure 2. Functional Block Diagram for ADV7123_a DAC Unit

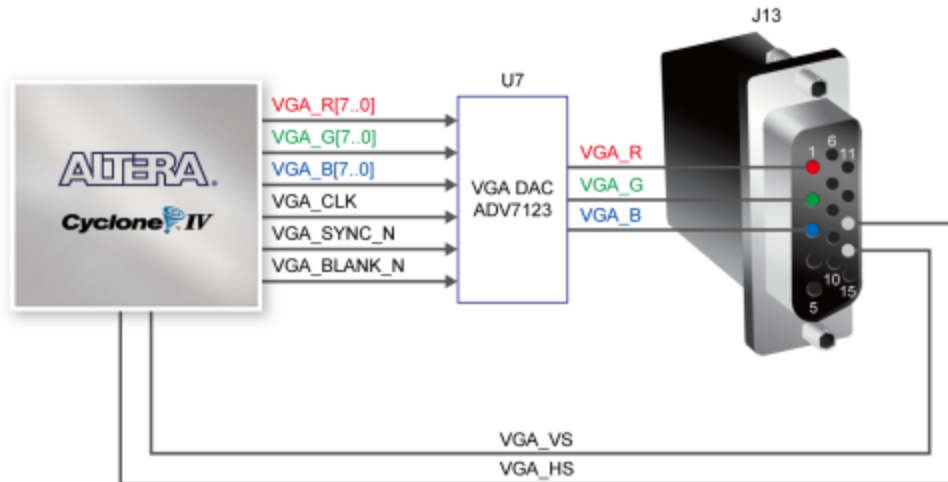


Figure 3. Functional Block Diagram for the DE2 VGA Subsystem

Throughout the rest of this report, the behavioral design and hardware implementation of the video interface will be outlined.

2 Behavioral Design

The behavioral design of the module was done in a rather quick and dirty manner in order to produce a working proof of concept. It uses one synchronous process and a sequence of if statements that check the values of the horizontal and vertical counters in order to determine output behavior. The module can easily be changed by using the generic inputs (which can be seen in Figure 4), and the output clock to the DAC unit can be seen updating asynchronously in Figure 5. The color data and blanking control work to produce the color signals during the visible intervals and make the DAC blank the color signals during the blanking intervals. The color values are increment based on the horizontal and vertical counters, and truncated to fit in 8 bits, in order to produce a repeating color range (which can be seen in Figure 6).

```
entity vga is
  -- constants obtained from http://tinyvga.com (for 800x600 pixel resolution)
  generic (
    h_pix      : integer := 800; -- # of horizontal pixels
    v_pix      : integer := 600; -- # of vertical pixels
    h_fporch   : integer := 56; -- horizontal front porch in pixels
    v_fporch   : integer := 37; -- vertical front porch in pixels
    h_pulse    : integer := 120; -- h synch pulse width in pixels
    v_pulse    : integer := 6; -- v synch pulse width in pixels
    h_bporch   : integer := 64; -- horizontal back porch in pixels
    v_bporch   : integer := 23; -- vertical back porch in pixels
  )
  port (
    h_synch    : out std_logic;
    v_synch    : out std_logic;
    blank      : out std_logic;
    color_data  : out std_logic_vector(23 downto 0); -- 24 bits of color data (8 bits per color in RGB order)
    vga_clk    : out std_logic;
    clk        : in std_logic;
  )
end vga;
```

Figure 4. Module's Entity Declaration

```
41 vga_clk <= clk;
42 vga_proc : process(clk)
43 variable h_count : integer range 0 to h_period - 1; -- using variables because they do not have to be
44 variable v_count : integer range 0 to v_period - 1; -- inputted/outputted anywhere else
45 variable red_count : integer range 0 to 255 := 0;
46 variable green_count : integer range 0 to 255 := 0;
47 variable blue_count : integer range 0 to 255 := 0;
48 begin
```

Figure 5. Module's Process and VGA Clock Feed-Through

```
77 -- controlling the blanking signal
78 if((h_count < h_pix) and (v_count < v_pix)) then
79     blank <= '1';
80     -- setting color to red for testing purposes
81     color_data(23 downto 16) <= std_logic_vector(to_unsigned(h_count*2, 8));
82     color_data(15 downto 8) <= std_logic_vector(to_unsigned((h_count - v_count)*2, 8));
83     color_data(7 downto 0) <= std_logic_vector(to_unsigned((255 - v_count)*2, 8));
84
85 else
86     blank <= '0';
87     color_data <= (others => '0');
88 end if;
```

Figure 6. Color Assignments and Blanking Control

3 Testing and Implementation

The implementation of circuit started without the use of color data. This was done in order to test if the nested-counter method of controlling the horizontal and vertical sync signals could work. As can be seen in Figure 7, the monitor is successfully able to detect the 800x600 pixel resolution the FPGA is trying to send it. This means the color data can now be sent.

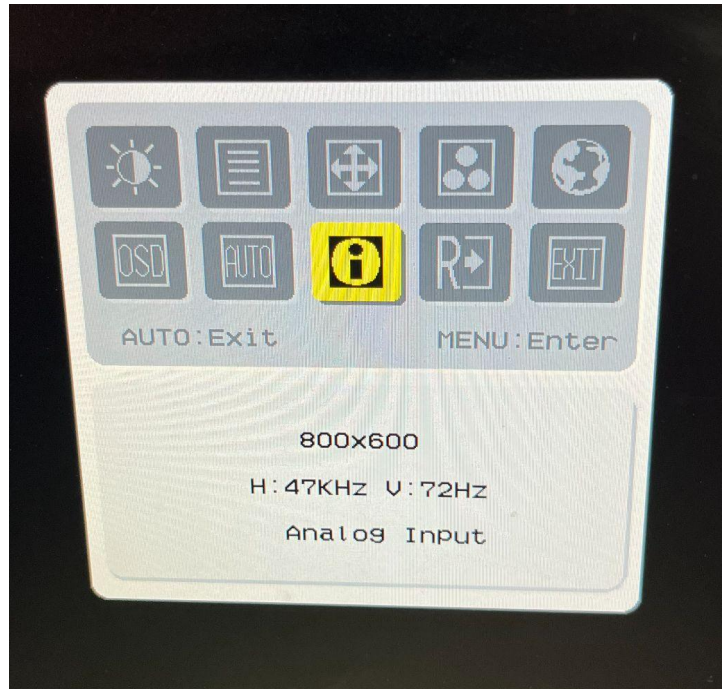


Figure 7. Monitor's Resolution Detection Feature

After the sync signals were demonstrated, the color data could then be sent. The color data, as mentioned previously, is updated based on the values of the horizontal and vertical counters, and the output signal can be seen on the monitor in Figure 8.



Figure 8. VGA Demonstration

After the resulting design was compiled in Quatus, in order to be deployed on the FPGA, the resulting register transfer level circuit and total number of logic elements were produced (which can be seen in Figures 9 and 10 respectively).

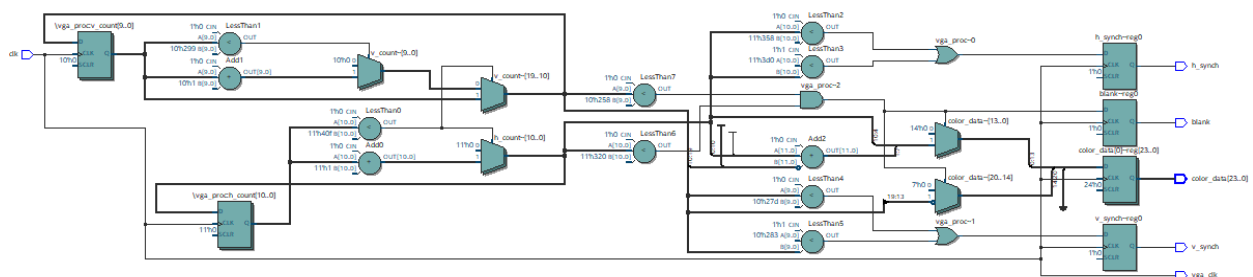
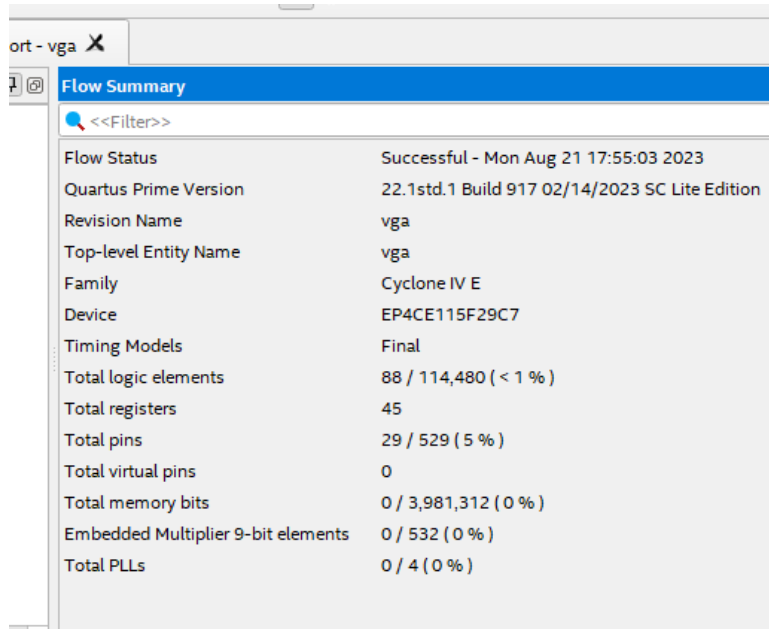


Figure 9. RTL View of Module



Flow Summary	
<<Filter>>	
Flow Status	Successful - Mon Aug 21 17:55:03 2023
Quartus Prime Version	22.1std.1 Build 917 02/14/2023 SC Lite Edition
Revision Name	vga
Top-level Entity Name	vga
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	88 / 114,480 (< 1 %)
Total registers	45
Total pins	29 / 529 (5 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 10. Quartus Compilation and Synthesis Results

4 Conclusion

Overall, this was a nice little side project to support my thesis. This implementation will hopefully help me make a more complicated proper video card that can read values from a piece of memory that is updated by the eventual CPU. Some of the limitations I hope to address in the future are the amount of memory needed in order to drive this resolution (perhaps a bit of external memory could be used), and if this resolution cannot be achieved, how may I scale the 50MHz clock in order to obtain the proper pixel frequency for a smaller resolution that uses less memory. Finally this project was a pretty smooth/straightforward process until I tried to add the color data. It took me a few hours of scanning documentation in order to realize that the pixel clock had to also be fed through to the DAC unit, and after that simple addition was made, the demonstration worked as initially intended. A further thing to look into may be “sync on green” mode which I discover in my reading, but did not have to implement because my monitor supports a separate green signal.

5 References

For general VGA timing constraints and frame section values:

<http://tinyvga.com/vga-timing>

DE2-115 Manual:

Pages 54 and 55

ADV7123_a Datasheet:

https://www.eecg.utoronto.ca/~tm4/ADV7123_a.pdf