

LAB 3: Video Output Interface

Abstract

The purpose of this lab is to learn how to display images and perform animation on a Video Graphic Array (VGA) component.

I. Introduction

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. For this lab, *VGA Pixel Buffer* is located in **SRAM** and *VGA Character Buffer* is located in the **on-chip memory**.

More information on the Video Cores may be found in *Video Out IP Cores for Altera DE Boards* [1].

II. Part 1: Testing the Provided DE2_70 Media Computer

The nios system has been generated based on the DE2_70_Media_Computer system [2]. A block diagram of the DE2-70 Media Computer system is shown in Fig. 1. A project for Altera Monitor Program has also been created for you in the `app_software` sub-directory. This Altera Monitor Program project links to the “`test_Media_Computer.c`” and file. This program tests various peripherals in the DE2_70 Media Computer.

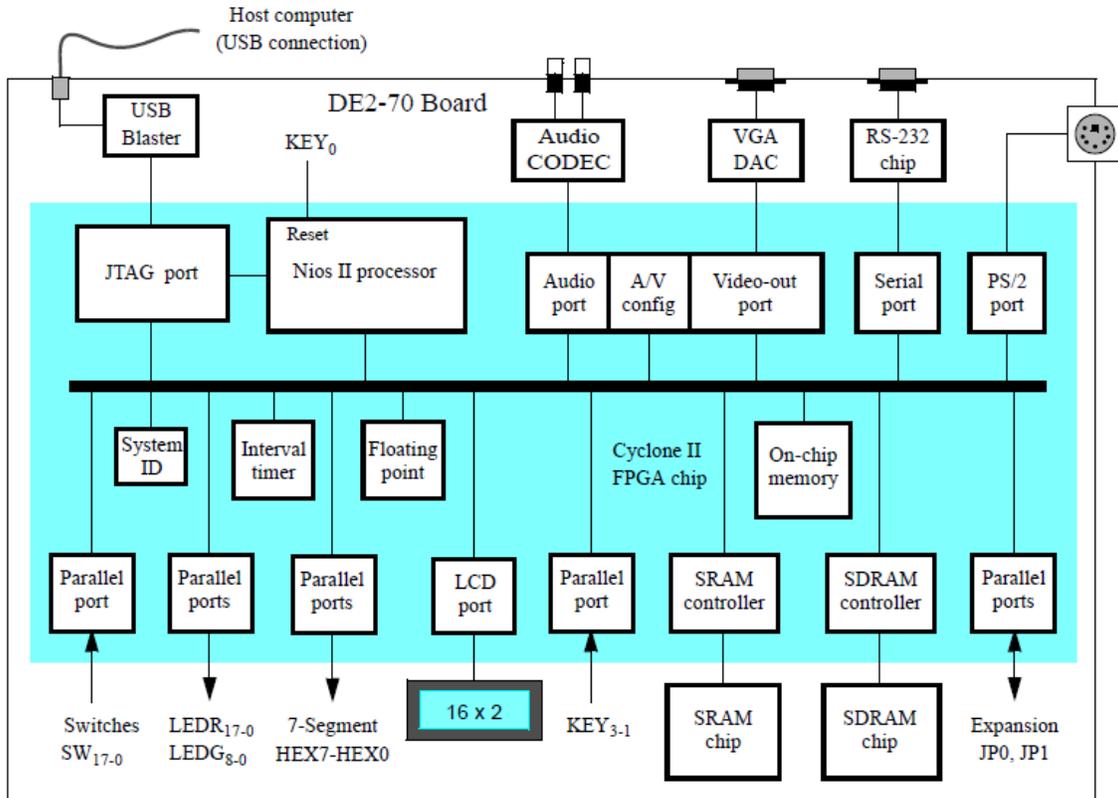


Fig. 1. Block diagram of the DE2_70 Media Computer.

Download the DE2_70_Media_Computer.zip from the syllabus page.

1. Review all the peripherals that are included in the nios system.
2. Review the provided “test_Media_Computer.c” and other supportive files.
3. Run “test_Media_Computer.c” on this hardware.

III. Part 2: Drawing an Object/Text

1. Write a C program to
 - a. create a black background screen,
 - b. draw a red square on the screen,
 - c. write “EC463 - Lab 3” on the screen
2. Demonstrate a working program.

IV. Part 3: Controlling VGA and LCD displays

1. Write a C program to read command from the jtag_uart and
 - a. move the square (part 2) **up** when the up arrow is pressed
 - b. move the square (part 2) **down** when the down arrow is pressed
 - c. move the square (part 2) **left** when the left arrow is pressed
 - d. move the square (part 2) **right** when the right arrow is pressed
 - e. display “EC463 - lab 3” on the first line of the 16x2 LCD
 - f. display “**left**”, “**right**”, “**up**” or “**down**” accordingly on the second line of the 16x2 LCD
2. Demonstrate a working program.

Note: Moving a displayed object is an illusion created by showing the same object at different locations on the screen. To move an object on the screen we must display it at one position first, and then at another later on. A simple way to achieve this is to draw an object at one position, and then erase it (by replacing an object with the background color) and draw it at another position.

V. Part 4: Graphic Animation

1. Write a C program to
 - a. draw a 4x4 box at a pseudo-random location on the screen,
 - b. start moving the box diagonally if the “space” key is pressed. The square should bounce off the top, bottom, left, and right edges of the screen. The box stops if the “space” key is pressed again.
2. Demonstrate a working program.

Note: The key to animation is timing, because to realize animation it is necessary to move objects at regular time intervals. The time intervals depend on the graphics controller. This is because the controller draws images onto a screen at regular time intervals. The *VGA controller* redraws the screen every 1/60th of a second. Since the image on the screen cannot change more often than that, this will be the unit of time.

To ensure that we draw on the screen only once every 1/60th of a second, we can use the *VGA pixel buffer* to synchronize a program executing the *nios_system* with the redraw cycle of the *VGA controller*. This is accomplished by writing 1 to the *Buffer register* and waiting until **bit 0** of the *Status register* in the *VGA*

pixel buffer becomes 0. This signifies that a 1/60th of a second has passed since the last time an image was drawn on the screen.

VI. Part 5: Implementing a Video Game

1. Write a C program to
 - a. implement a *PONG* game. I
 - b. include the following options:
 - i. Display the current point on the screen
 - ii. Maintain and display the High game
 - iii. Increase the speed after each checkpoint (i.e. increase one speed step every 10 points).
2. Demonstrate a working program.

VII. Bonus: Displaying with a LTM touch screen

1. Modify the nios_system to output the video to the LTM display.
2. Implement the PONG game on the LTM display.

VIII. Reference

- [1] Altera, "*Video Out IP Cores for Altera DE Boards*," 2009.
- [2] Altera, "*External Clocks for Altera DE Boards*," 2008.