**Synplicity/ModelSim/Xilinx Tutorial II and Lab Exercise**
**Using Internal ROM in Xilinx Spartan-3 devices**

**University of California, Davis**
**Department of Electrical and Computer Engineering**

**Objective:** This tutorial explains how to implement ROM modules in the Xilinx Spartan-3 device. In this tutorial, we will design a character-based video controller by following the examples given in Chapter 9, sections 9.8 – 9.10, of Rapid Prototyping of Digital Systems, by Hamblen and Furman. We will illustrate how ROM can be specified, simulated and implemented using the Synplicity, ModelSim, and Xilinx design tools.

## I. Overview

There are two basic ways that you can implement a memory module such as a ROM or RAM in a Xilinx Spartan-3 device. You can either *inference* the memory module by writing your VHDL code in such a way that the synthesis tool understands that your behavioral code is specifying a memory module, or you can *instantiate* a memory component from a technology-specific library, in this case a Xilinx library. In this tutorial, we will focus on inferencing memory devices since it has the advantage of being technology-independent. For information on instantiating memory devices, see Xilinx's Application Note XAPP463, "**Using Block RAM in Spartan-3 Generation FPGAs**", which is available on the course web page. For information on inferencing RAM devices, see Synplicity's Application Note, "**RAM Inferencing in Synplify Software Using Xilinx RAMs**", which is also on the course web page.

The Xilinx Spartan-3 FPGAs contain block RAM memories that can be used for large, on-chip memories. The XC3S200 device that is on our target board, for example, contains twelve 18-Kbit RAM blocks. The RAM within the 4-input lookup tables (LUTs) of the Spartan-3 device can also be used to implement small distributed memories using 16x1 RAMs.

## II. ROM Inferencing for Character Display

In this example, we will store a set of character fonts in a ROM. We will store the 64 characters listed in Table 9.1 of the Hamblen and Furman text using the 8x8 pixel maps detailed in the tcgrom.mif file, which comes with the text. A 512x8 ROM is sufficient to store the 64 8x8 character fonts.

In order to complete this tutorial, create a new directory and copy the files from the /afs/ece/classes/eec180b/xilinx_rom_ex directory into your new directory. Study these 4 VHDL source files (char_rom.vhd, vga_sync.vhd, char_test1.vhd, and tb.vhd). Notice that in char_rom.vhd, we have used a giant constant array to infer a ROM device. When we synthesize the design, we can check the RTL view to verify that Synplicity actually does synthesize this structure as a ROM. The code in vga_sync.vhd is essentially the code in Ch. 9 of Hamblen and Furman. (Study this chapter to understand how this

module works.) The char_test1.vhd code is the top-level design, which makes use of the char_rom and the vga_sync modules.

- Simulate the character-based video controller, char_test1, using the testbench provided. Simulate until you see some activity on the **red_out** signal. Determine the pixel row, pixel column, and the character being displayed when the **red_out** signal first goes high.

- Synthesize the design using Synplicity's Amplify tool. Open the RTL View (.srs file) and verify that the char_rom module has been synthesized as a ROM. Look at the vga_sync module as well.

- Run the Xilinx ISE tool to generate a bit file from the edf file produced by Amplify. Look in the MAP Report to see if Block RAM or distributed LUT memory such as 16x1 ROMs are used to implement the character ROM.

- Download the bit file to the Xilinx board and verify the operation.

It is very important that you understand how this design works because you will be using several of these modules in your other labs such as the graphics display lab and the CPU lab.

## III. Lab Requirements

To successfully complete this tutorial, perform the following tasks:

- Modify the design to display the characters using 16x16 pixels rather than 8x8 pixels. (Note: You do **not** need to change the char_rom module!) Also, change the color scheme so that the rows are divided in fairly equal blocks of the 7 non-black colors. In other words, instead of changing color each row, change the color after each approximately (1/7) of the rows. As an option, you could change colors based on the column instead of the row. Simulate, synthesize and download your new design and demonstrate it to your TA.

- Have the TA verify your simulation and your working circuit and sign a verification sheet. There is no Lab Report required, but you must turn in the signed verification sheet.