RetroMicro FPGA Experiments

...designs for the computer hardware enthusiast...

www.RetroMicro.com

eMail: doug@RetroMicro.com

The book titled ***"Rapid Prototyping of Digital Systems - A Tutorial Approach by James O. Hamblen and Michael D. Furman"*** is an excellent introduction to programmable logic, VHDL and logic synthesis. The book provides a wealth of design examples that provide a great foundation for the computer hardware enthusiast designing microprocessor-based systems.

Altera's MAX+PLUS II design environment was used to create all the designs contained in the book. While this is a good design environment, I have personally found that the newer Quartus environment to be even better and cleaner in implementation. Also, Altera recommends new designs be created in Quartus.

Provided in this zip archive is all of the MAX+PLUS II code slightly restructured and moved into equivalent Quartus projects. In a few cases, Quartus would not compile the code as is; Quartus seems to have a few tighter constraints in a number of areas. All code issues have been worked out as discussed below.

The projects in this archive have been organized as follows:

Project Directories	**Description**
upLib	Associated with many projects. This is a library of the core modules for video, keyboard, mouse, debounce, etc... circuits. See chapter 5 for brief module descriptions.
tutorial\tutor2	Chapter 1-4: Simple counter that displays the count value on the 7 segment LEDs. This example circuit illustrates a basic problem with interfacing to external I/O (a push button in this case).
tutorial\tutor3	Chapter 1-4: Simple counter that fixes the problem above with a clock divider and debounce circuit.
train	Chapter 7: It draws a "train" moving around a track on your VGA monitor. This is an elaborate VHDL example on how to design a state machine.
uP1	Chapter 8: A simple CPU design. This code does not use the board, but can be simulated.
vga\ball	Chapter 9: Draws a ball moving vertically and bouncing ball off the "edges" of the monitor.
vga\char_test	Chapter 9: Draws a bunch of characters on the monitor using a character "ROM".
vga\vga_bar	Chapter 9: Draws a series of colored "bars" on the monitor.
vga\vga_led	Chapter 9: Uses the push button to toggle the color drawn on the monitor.
vga\vga_test	Chapter 9: Draws a crosshatched colored pattern on screen.
keyboard	Chapter 10: Simple test of the keyboard module. Displays received scan codes on 7 segment LEDs.
mouse	Chapter 11: Simple test of mouse module. Displays received mouse information on 7 segment LEDs.
mips\computer	Chapter 13: A full implementation of the MIPs cpu interfaced with video. Designed to be downloaded to UP2 board. Push buttons are used to control execution.
mips\cpu	Chapter 13: MIPs implementation designed for simulation and testing of the core cpu. Good starting point for the pipelining lab.
lc-2	Extra project included on CD but not discussed in book. A nice implementation of the lc-2 cpu presented in the book titled "Introduction to Computing Systems: From Bits & Gates to C & Beyond".

General Notes

Project location – All projects should be placed in the c:\qdesigns directory. If placed elsewhere, library and file paths may have to be modified in order to compile correctly.

Project structure – The “source code” for each project is located in the directory titled “src” inside each project directory. This was done to explicitly keep original source code away from all the other Quartus generated project information and synthesis files. Original hand drawn block diagram (schematic) files are also stored in the src directory. Since Quartus generated symbol files can easily be recreated from the original HDL source file, they are usually left in the main project directory.

Quartus versions – Sometimes when opening a project file created with a different version of Quartus results in a warning about different database formats. Clicking “ok” will allow Quartus to convert it to the version you are using (either newer or older). We have not experienced any problems allowing Quartus to do the conversion, if necessary.

UP2 board – All the projects have the targeted device set to the FLEX10K CPF10K70C240-4. This is the device on the UP2 board. If you are targeting a different board/chip, make sure to set the proper target device.

Project Specific Code Changes

upLib : This project is really the upcore library presented in chapter 5 of the book. It was renamed (for no particularly good reason) to upLib. This is the only “project” in which original source code is stored in the “root” project directory. The reason for doing this has to do with the fact that upLib is a library and not really compiled by itself. Another reason for keeping the original source code in the same directory as the block symbol files has to do with convenience; it reduces the number of directories that need to be specified when including components. Also note that a package file called upLib.vhd has been created to make it easy to include library components into a design. Note: The only reason there is a project file associated with these components is for symbol file creation.

uP1 – The primary change to this project entailed a “fix” so that Quartus could synthesize it. Quartus has a problem in that you cannot specify a MIF (Memory Initialization File) file if either data input or address lines are specified as “UNREGISTERED”; both must be specified as “REGISTERED”. The design itself could be changed to work with registered memory correctly, but it was found that the problem could be solved another way. Since there is a need to access program instructions, the solution divides up memory so that is has two components, RAM and ROM. ROMs allow MIF file to be specified even if address lines are specified to be “UNREGISTERED”. A simple address decoder was added to the design to facilitate this memory design.

mips – A few changes in the basic structure of both of the mips oriented projects have been made. The first is an actual renaming of the top entities from “top_spim” to “cpu” and “top_flex” to “computer”. Another change is an explicit creation of a “cpu” entity. The only purpose of the “cpu” project is to synthesize the cpu and simulate it’s behavior. The “computer” project simply interfaces the cpu to a full up system that can be run on the board. Common source code for both projects is shared in the mips/src directory. Memory in the dmemory module was also divided for the same reasons as discussed above.