Experiment 3

Development of Software Troubleshooter for faulty Wimp51

**INTRODUCTION:**

In the last lab, you were given a software specification from which to write a program and were given a model of the Wimp51 processor on which to prototype your design using Altera DE-2 FPGA board. After completely verifying your program through prototyping, you are now ready to troubleshoot in case of any problems with the hardware. The troubleshooter will be developed in this lab for a faulty Wimp51 model. The fault will be detected using DE-2 board and the multiplication code will be modified as a work-around.

**OBJECTIVES:**

1. Familiarization with the Altera DE-2 Board
2. Understand tradeoffs between the two approaches of testing: software simulation and the use of actual hardware
3. Modify software to “fix” an unexpected change in the hardware
4. Recognize internal behavior of a processor by observing external signals

**MATERIALS REQUIRED:**

- Altera DE-2 FPGA board
- Windows-based Computer with an unused USB port
- Altera Quartus-II software suite
- Wimp51 model with an injected fault

**BACKGROUND:**

After software simulation is complete, software should be verified with actual hardware before full-scale system production. Without hardware verification, there is no guarantee that the software will function properly in the final product. Several problems might prevent a successful implementation. Hardware designers might modify their design mid-way through the design process without properly informing the software engineers. The simulation might have been conducted with an old hardware model. The model might have errors. The hardware may have quirks that were not accounted for in the hardware model. Any of these events could effect whether the software will function correctly. Despite the need to verify your design in hardware, there are several advantages to verification through simulation. In simulation, you have complete control of your environment and many signals may be available to you that are not available in hardware.
For example, during simulation you can run your design for 100mS, stop everything to analyze the results, and then start back up again where you left off. Stopping and starting the actual hardware may not be so easy. Your simulation tool might also give you easy access to internal registers, like the accumulator, program counter, or instruction register, which makes debugging much easier and faster. At first glance, simulation will seem like an unnecessary and time-consuming task. With experience, though, you will find that using a healthy dose of simulation to test your design is far better than the “burn and try” approach, where code is immediately burned into memory just to see what happens. The advantages of simulation are especially apparent in large projects. The least expensive method to verify hardware is through a prototyping board such as the DE-2 board by the Altera. Production of a custom hardware prototype, fabbed on its own PCB board possibly with custom Application Specific Integrated Circuits (ASICs), could cost thousands of dollars and would be wasted money if the design failed. The costs are even higher if the entire design is to be implemented on a single chip (called Systems on Silicon (SoS) or Systems on a Chip (SoC).

**PROCEDURE:**

1. Copy all the contents files of Lab2 except for the EDIF model of Wimp51. The Wimp51 with an injected fault will be provided to you by your instructor.

2. Create a new project in Quartus-II named Lab3, and include all the necessary EDIF, VHD and BDF files in it. You can also import the pin-assignment of Lab2 to maintain the consistency. Create a symbol for Wimp51 with injected fault.

3. The top level entity of this project is exactly the same as that in Lab2, except that the symbol for Wimp51 is replaced by the one with injected fault.

4. Compile the project with pin assignments. Under normal circumstances, the compilation should go through without any errors. If you have errors, they would normally be due to conflicts in pin assignments or schematic creation. Fix errors of any, and recompile the project.

5. After the compilation, download the design on the DE-2 FPGA board using programmer tool in Quartus-II.

6. Now observe the output of seven-segment displays with every clock edge and locate the execution point where it shows an unexpected behavior.

7. You can also use the Simulation tool in Modelsim to verify and locate the fault injected. Refer to Appendix A to know more about simulation in Modelsim.

8. Consult with your instructor about the fault you noticed in the given Wimp51 model.

9. After the fault is successfully detected, make the necessary changes in the assembly code for the multiplication of two numbers. Obtain the corresponding HEX code for the new code, and make the changes in the ROM module of your project. Note that due to these changes, you will need to update your ROM symbol, and recompile the entire design.
10. After the compilation, download the design on DE-2 FPGA board, and find out whether the change in the code gives the desired results, i.e. those obtained in Lab2. Redo steps 9 and 10 until desired results are obtained.

**QUESTIONS:**

1. What fault did you detect in the given Wimp51 model?

2. Explain in brief, how this fault affected the execution of the code

3. What changes were made in the code to work around this fault?

4. Inclusion of which instruction in the instruction set would have been the shortest work-around?

5. Did the change in the design change the number of logic blocks used in FPGA