Using the Visual Wimp51

The following is an excerpt from D.G. Beetner and H.J. Pottinger, *CmpE213 Computer Engineering Laboratory Manual*, Barnes and Noble Custom Publishing Series, 2001,"EXPERIMENT NUMBER 2: Introduction to Hardware Software CoSimulation." It will give you introductory information on using the Visual Wimp51.

Background

To be useful to a microprocessor, programs must be broken down into machine code, something the processor can understand. Machine code is usually generated by a compiler or an assembler. This machine code must be translated into a file-format that can easily be transferred to the processor or to external memory. A standard file that many loaders use is the Intel hex-file format. The following is an example of a program in this format:

```
:10008000AF5F67F0602703E0322CFA92007780C361
:1000900089001C6B7EA7CA9200FE10D2AA00477D81
:0B00A00080FA92006F3600C3A00076CB
:00000001FF
```

See your textbook for more details of the Intel hex-file format.

To convert from assembly language to machine code, we use a program called an assembler. This program converts your assembly language file into a binary object file format. Another program will convert this object file into Intel hex-file format.

For example, the set of commands to use with the Tasking 8051 assembler (available on our Sun workstations) is
asm51 filename.a51
ihex51 filename.obj filename.hex

And the set of commands to use with the Keil Software 8051 assembler (available on the Windows NT stations) is:
A51.EXE filename.a51
OH51.EXE filename.obj HEXfile (filename.hex)

Using the Visual Wimp

1. You may create a new directory called lab2 or use your old lab1 directory for this experiment. Download a new XS40 simulation model into this directory from http://www.ece.umr.edu/courses/cpe214/dist/lab2.tar. Untar the downloaded model within this directory using the "tar –xvf" command, as given in Lab 1.
2. Set the working directory and user library using the commands `setenv MGC_WD `pwd`` and `setenv USERLIB `pwd`` within the directory you are using for Lab 2.
3. Build the simulation models you will need using the commands:

```
build_simulation
build_wimp
```

These files are executable scripts, which we have written to run several Mentor Graphics commands for you. The first command compiles several VHDL files used in the XS40 board simulation model. The second command compiles the model of the Wimp51 processor.

4. To load your program into the simulation model, copy your Intel hex file (created as part of the preliminary assignment) to the file $USERLIB/program_files/sram.hex. The best way to do this is by creating a symbolic link to your file. For example, you might type:

   ```
   ln -s program.hex $USERLIB/program_files/sram.hex
   ```

   assuming your program, program.hex, is in the same directory as the symbolic link. This command creates a link named sram.hex which points to the file program.hex. For all practical purposes, sram.hex and program.hex are the same file. The simulation model assumes the file sram.hex contains the contents of the Wimp51's program memory. By copying (or linking) your file here, you are effectively loading Wimp51 memory with your program.
5. You are now ready to simulate your program on the Wimp51 simulation model. Simulation will be done with Mentor Graphics QuickSim Pro. QuickSim Pro, or QSPro for short, combines the Quicksim program (used for simulating schematics) and the Vsim program (used for simulating VHDL models). Start QuickSim Pro using the command:

   ```
   qspro –as hdl xess40_schematic
   ```
6. Simulate and verify your Wimp51 program:

a. Reset the Wimp51 and provide it a clock signal by applying a negative pulse to the reset line (J1-3) and a square wave to the clock line (J1-2), as explained below.
   - Open the XS40 schematic sheet by choosing File, Open Sheet at the menu at the top of the QuickSim window.
   - Deselect the address bus, data bus, and control lines (if selected) by hitting "F2". Notice that the function keys functions are given at the bottom of the window.
   - Select both the reset and clock lines in the XS40 board schematic, click 'WF Editor', and click 'Edit Waveform'. Both signals are now added to the trace listing as force signals. Note that you should always force portin or ipad components. Never apply forces to a wire.
   - Use the force command to create a 5 ns wide negative pulse starting at 5 ns for the reset line. To begin, type the following command into the command window:

     ```
     force J1-3 1 0
     ```

     To activate the command window, just type when the sheet (your design) is highlighted. In this command, 'J1-3' is the signal you are forcing, '1' is the level you are forcing it to, and the '0' is the time (in ns) at which it will be forced. So, signal J1-3 will become a logical 1 after 0 ns of simulation time. Now that we've ensured the line begins at a logical 1, we want to force the line to a logical 0 at 5 ns and 1 at 10 ns.

    Add the last two forces necessary to give a 5 ns wide negative pulse starting at 5 ns.

  - Create a clock on J1-2 the same way you did in experiment 1. Use a period of 25 ns.

d. Up until this point you have been doing everything in the Quicksim window. Now let's explore the other window that appeared when you started QSPro. The window should be labeled QSPro (HDL) or just Vsim. You can type Tcl-based commands into this window to examine and debug the VHDL portion of the model. Tcl is a special language used to develop QSPro. One advantage to using a simulation model, over a hardware prototype, is that special features can be built in that help debug problems. In this step we will look "inside" the Wimp51 to see the values of internal registers.
  - Type "source wimp51.tcl" at the prompt in the Vsim window. This command tells Vsim to read in a series of Tcl-based commands from the file "wimp51.tcl." You will see a window appear that shows a block diagram of the internals of the Wimp51 processor.
  - The "Next clock" button allows you to step through your program one clock cycle at a time. You can see the data flow through the inside of the processor for each clock step. Any time the value of a register changes, that register's value will be highlighted in red for one clock cycle. The current decoded instruction appears in the control unit window.
  - The "Next instruction button" allows you to step through your program one instruction at a time. While you are using either of these two buttons, the Quicksim window will be recording the signals traces from your program in the trace window.
  - You may use the "Reset button" to restart your program from the beginning. Note that the signal traces in the Quicksim trace window will be erased if you do this.
  - Step through your original assembly program and make sure the instructions you see in simulation match what you expect to see If it does not match, you have probably made a mistake in forming your hex file. If this is the case, you will need to fix your hex-file and start the simulation over again.

    If your program has errors, you may edit the hex file and recopy it to sram.hex through a separate command window, then reset and re-run the simulation (as in lab 1). You do not need to exit and restart QSPro after you change the hex file.

   Continue stepping through your program until you are satisfied that it is working properly.