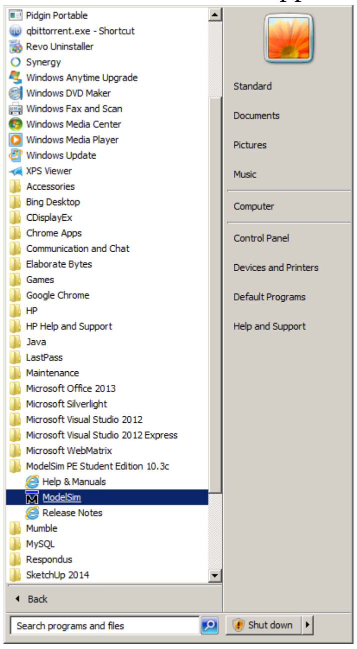
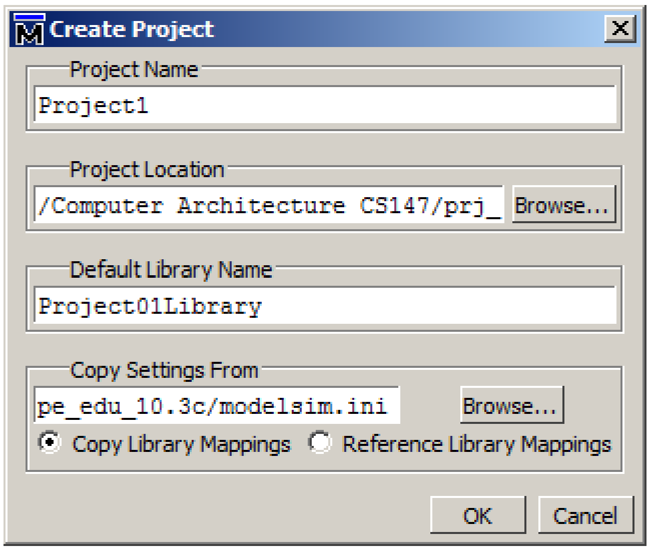
ModelSim Installation and basic use with Verilog operators

D. Thorpe

1. Introduction- This document is to represent my work done on the installation and basic use of ModelSim as well as a basic program using Verilog operators.
2. Steps on how to install the simulation (10.3c) tool, Windows.
   2.1. Download the application installation file for ModelSim PE Student Edition currently maintained and distrusted by Mentor Graphics at http://www.mentor.com/company/higher_ed/modelsim-student-edition
   2.2. Ensure the desired user is logged in and has administrative rights to perform installations. UAC elevation can cause installation through alternative accounts, generating license issues further down the line.
   2.3. Run installation file based on your system configuration. Be sure the installation directory has sufficient permissions to allow the application to run unhindered, the "Program Files" directory may be unsuitable for this requirement.
   2.4. Apply the required changed to your Path variable based on system requirements and configuration.
   2.5. After installation complete and submit the required license request form to Mentor Graphics and after receipt of license file download to the root level folder of the ModelSim installation.
   2.6. Problems during installation are best researched via https://groups.google.com/forum/#!forum/modelsim-pe-student-edition
3. Steps of simulation project creation (including screen shots), Windows.
   3.1. Launch the ModelSim application.



   3.2. Create a new project via File>New>Project and complete fields with unique relevant information.


Create Project

Project Name: Project1

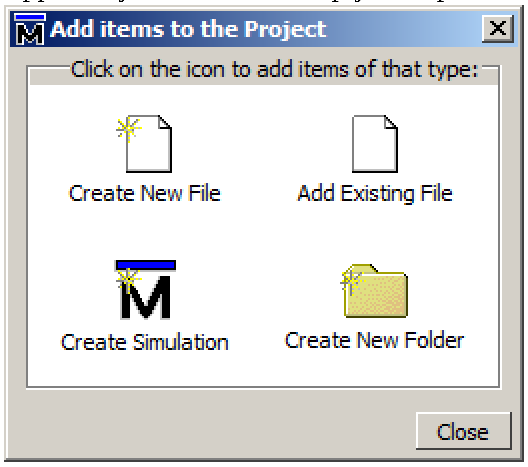
Project Location: /Computer Architecture CS147/prj_

Default Library Name: Project01Library

Copy Settings From: pe_edu_10.3c/modelsim.ini

[✓ Copy Library Mappings] Reference Library Mappings

3.3. Add required files and directories to the Project via the next prompt. This are the files supplied by the Professor in prj_01.zip.


Add items to the Project

Click on the icon to add items of that type:

3.4. Select close. The simulation project is new created.

4. Requirement, 32 bit combinatorial ALU Supports the following functions:
   4.1. Integer add (0x1)
   4.2. Integer sub(0x2)
   4.3. Integer mul(0x3)
   4.4. Integer shift_rigth (0x4)
   4.5. Integer shift_left (0x5)
   4.6. Bitwise and (0x6)
   4.7. Bitwise or (0x7)
   4.8. Bitwise nor (0x8)
   4.9. Bitwise set less than (0x9)
5. Design and implementation of ALU:
   This is done two fold in the prj_01_tb.v and alu.v within the specified TBD area in the Verilog language.
   5.1. The following lines of code were inserted into prj_01_tb.v, replacing SIGN and # with the proper operator and number:
   `ALU_OPRN_WIDTH'h# : begin $write("SIGN "); golden = op1 SIGN op2; end
   5.2. The following lines of code were inserted into alu.v:
   `ALU_OPRN_WIDTH'h# : result = op1 SIGN op2;

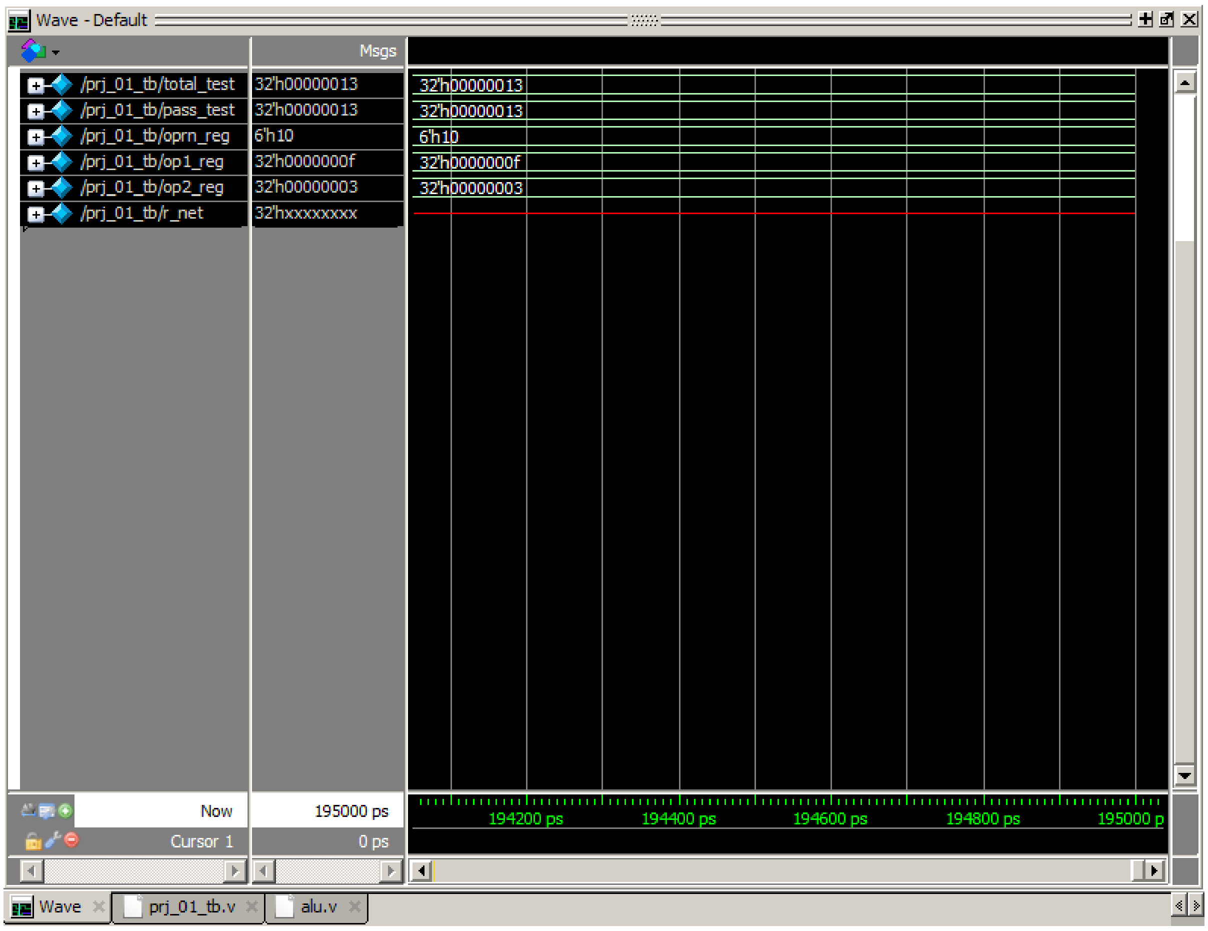
   #5 op1_reg=#;

```
        op2_reg=#;
        oprn_reg=`ALU_OPRN_WIDTH'h#;
    #5  test_and_count(total_test, pass_test,
                       test_golden(op1_reg,op2_reg,oprn_reg,r_net));
```

6. My Test strategy was implemented with two simple tests for each.
   6.1. Text output:

```
run -all
# [TEST] 15 + 3 = 18 , got 18 ... [PASSED]
# [TEST] 15 + 5 = 20 , got 20 ... [PASSED]
# [TEST] 15 - 3 = 12 , got 12 ... [PASSED]
# [TEST] 15 - 5 = 10 , got 10 ... [PASSED]
# [TEST] 15 * 3 = 45 , got 45 ... [PASSED]
# [TEST] 15 * 5 = 75 , got 75 ... [PASSED]
# [TEST] 15 << 3 = 30 , got 30 ... [PASSED]
# [TEST] 15 << 5 = 30 , got 30 ... [PASSED]
# [TEST] 15 >> 3 = 7 , got 7 ... [PASSED]
# [TEST] 15 >> 5 = 7 , got 7 ... [PASSED]
# [TEST] 15 & 3 = 3 , got 3 ... [PASSED]
# [TEST] 15 & 5 = 5 , got 5 ... [PASSED]
# [TEST] 15 | 3 = 15 , got 15 ... [PASSED]
# [TEST] 15 | 5 = 15 , got 15 ... [PASSED]
# [TEST] 15 ^& 3 = 0 , got 0 ... [PASSED]
# [TEST] 0 ^& 0 = 0 , got 0 ... [PASSED]
# [TEST] 15 | 3 = 0 , got 0 ... [PASSED]
# [TEST] 1 | 1 = 0 , got 0 ... [PASSED]
# [TEST] 15 ? 3 = x , got x ... [PASSED]
#
#       Total number of tests       19
#       Total number of pass        19
#
# ** Note: $stop    : C:/Users/Standard/test/prj_01_tb.v(154)
#    Time: 195 ns  Iteration: 0  Instance: /prj_01_tb
```

   6.2. Wave output:



```
Wave - Default
                         Msgs
/prj_01_tb/total_test    32'h00000013    32'h00000013
/prj_01_tb/pass_test     32'h00000013    32'h00000013
/prj_01_tb/oprn_reg      6'h10           6'h10
/prj_01_tb/op1_reg       32'h0000000f    32'h0000000f
/prj_01_tb/op2_reg       32'h00000003    32'h00000003
/prj_01_tb/r_net         32'hxxxxxxxx

Now          195000 ps      194200 ps   194400 ps   194600 ps   194800 ps   195000 p
Cursor 1     0 ps
```

7. In Conclusion the ModelSim application is an incredibly complex tool of which I have only gain a crude understanding. The Verilog language thus far, though, has not been far cast from my experience in C and Java.