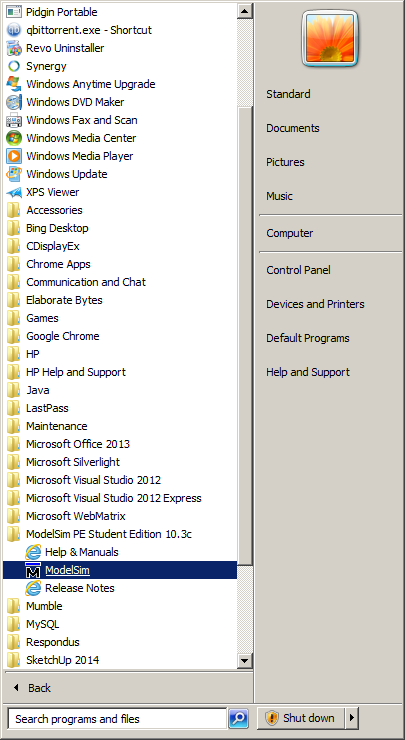
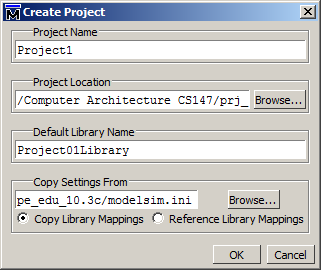
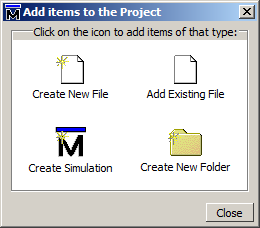
**ModelSim Installation and basic use with Verilog operators**

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.
   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. 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.
   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.
   4. Apply the required changed to your Path variable based on system requirements and configuration.
   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.
   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.
   1. Launch the ModelSim application.  
       
   2. Create a new project via File>New>Project and complete fields with unique relevant information.



* 1. Add required files and directories to the Project via the next prompt. This are the files supplied by the Professor in prj\_01.zip.  
      
  2. Select close. The simulation project is new created.

1. Requirement, 32 bit combinatorial ALU Supports the following functions:
   1. Integer add (0x1)
   2. Integer sub(0x2)
   3. Integer mul(0x3)
   4. Integer shift\_rigth (0x4)
   5. Integer shift\_left (0x5)
   6. Bitwise and (0x6)
   7. Bitwise or (0x7)
   8. Bitwise nor (0x8)
   9. Bitwise set less than (0x9)
2. 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.
   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
   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));

1. My Test strategy was implemented with two simple tests for each.
   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

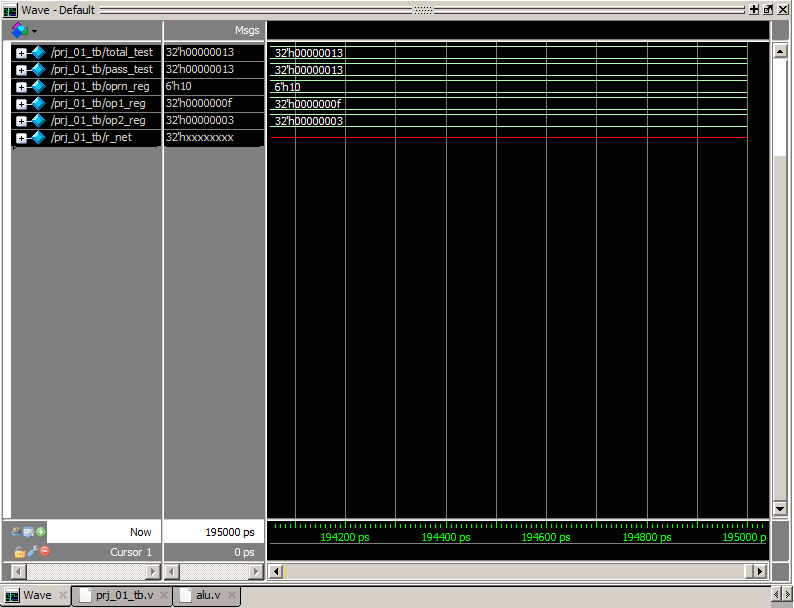
# Total number of pass 19

#

# \*\* Note: $stop : C:/Users/Standard/test/prj\_01\_tb.v(154)

# Time: 195 ns Iteration: 0 Instance: /prj\_01\_tb

* 1. Wave output:



1. 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.