Lab Exercise 2:

Investigating what is going on behind the scenes with the VHDL compiler

Objective: Be able to use the modelsim environment independently of a targeted IDE.

Procedure:

1. Setup you working environment.
   Create a directory for you to do your work in, if this is a new directory than we need to create a working library for our designs to compile into.
   From a cmd shell, change to this directory and create a VHDL library called *work*:
   vlib work
2. Run vcom and examine the possible parameters.
3. Using a text editor of your choice, but not the Xilinx ISE write the following VHDL codes.
   a. Write a full adder that uses a half adder as an instantiated component. This full adder is to be accomplished in a single VHDL file. That is your single file should contain multiple entity declarations and the associated architecture declarations. (see page 72).
   b. In a new file, Write a half adder that takes two one bit inputs and produces a one bit sum and one bit carry. (see page 65)
   c. In a new file, Write a or_2 that takes two one bit inputs and produces a one bit output that is the or of the two inputs. (see page 65)
   d. In a new file, Write a second full adder that instantiates the half adder and or_2 from your local library and produces the full adder function. (see page 74)
4. After each step in part two, save your vhdl code to your working directory. From the command prompt issue the command:
   vcom filename.vhdl
   If you are having trouble with missing signals and named process getting combined. You may want to try
   vcom +acc filename.vhdl
   or
   vcom -novopt -O0 filename.vhdl

5. Simulate each of the designs your produced in part 3. (4 simulations in total)
   From the command prompt in your working directory launch modelsim
   vsim entity_name

You are to turn in a write-up similar to a formal lab report.

Useful commands during simulation:

add wave signal_name - adds a signal to the wave window, opens a wave window if necessary.

add wave entity_name/signal_name - adds a signal from a specific entity to the wave

	window. This cab be useful when working with hierarchal designs and wishing to view signals from embedded design elements
add wave * -	adds all signals in scope to wave window
add list xxxx -	works like add wave only put signals in the list window
force signal_name value	forces the named signal to the named value at the current position in time
force signal_name value time	forces the named signal to the named value at the named time in the future
force -freeze xxxx	forces the signal to an unchangeable value
force -deposit xxxx	forces the signal to an initial value that will be changed the next time the signal is updated
force - drive xxxx	forces the signal to a value that will be resolved with the next update that is scheduled for the signal

force or_2/b 1 0, 0 {50 ps} -r 100

this is a clock forced onto a signal line
this statement actually says force b to 1 now and 0 in 50 ps
then repeat this pattern every 100 ps
this produces a 50% duty cycle clock with a period of 100 ps

run time	this cause time to progress forward for named time
do filename	this executes a macro do file