

HOMWORK – USING MODELSIM

Prof. Don Bouldin – 20 March 2008

Purposes: Perform pre-synthesis simulation using *ModelSim*.
(a) Test a logic component (fulladd.vhd) interactively.
(b) Use a script to test a structural combination of multiple components.

Part A – Tutorial

Download:

<http://www.ece.utk.edu/~bouldin/protected/551-hw2.tar.gz>

or copy the files on ada0.ece.utk.edu: `cp ~bouldin/webhome/protected/551-hw2.tar.gz .`

gunzip 551-hw2.tar.gz

tar -xvf 551-hw2.tar

Now, move down to the subdirectory:

1. **cd 551-hw2**

Then, prepare a working directory for *ModelSim*:

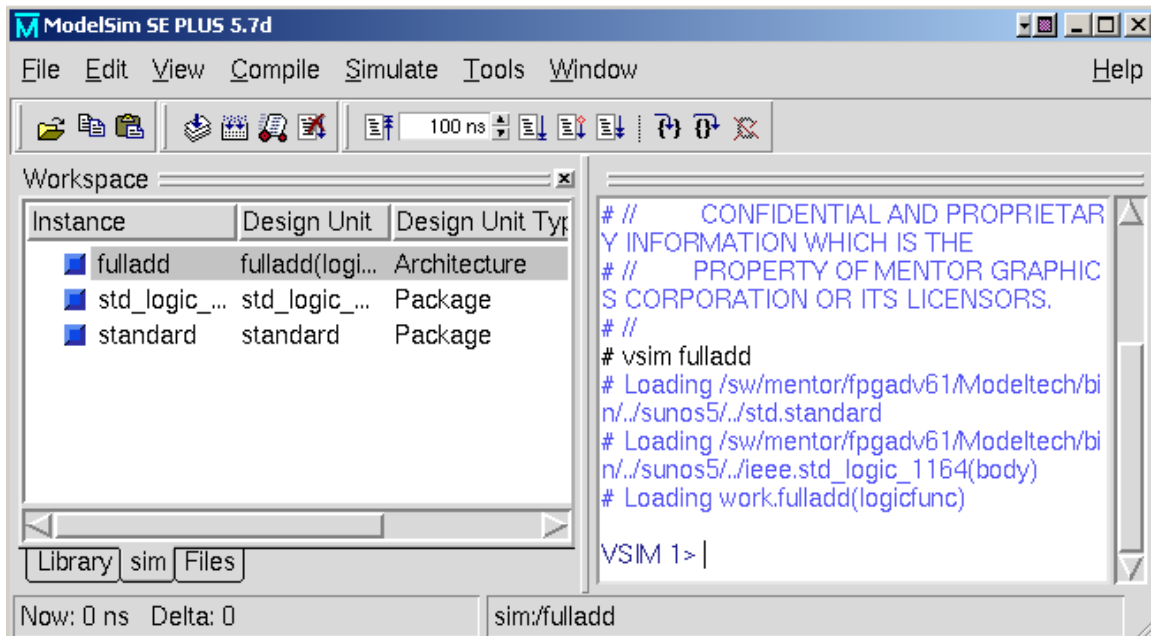
2. **vlib work**

Next, compile and simulate the component file for the full adder:

3. **vcom fulladd.vhd**

4. **vsim fulladd**

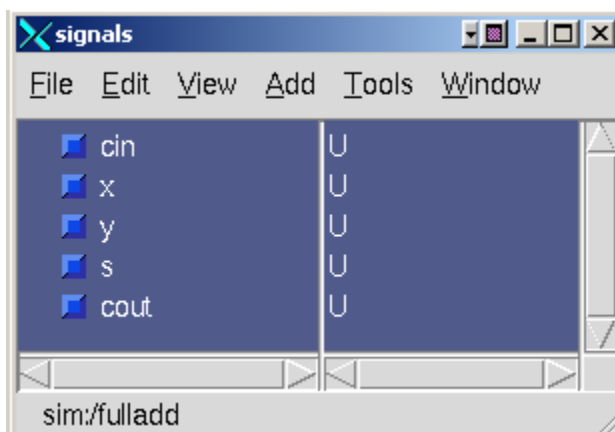
This will bring up the following window (Version 5.7d):



From the pull-down menu, use the left-mouse-button to select:

5. View→Signals

This will bring up the following window:



You can select individual signals by positioning the mouse arrow over any of the signal names and clicking the left-mouse-button to highlight it.

Alternatively, you can select all of the signals by using the pull-down menu to select:

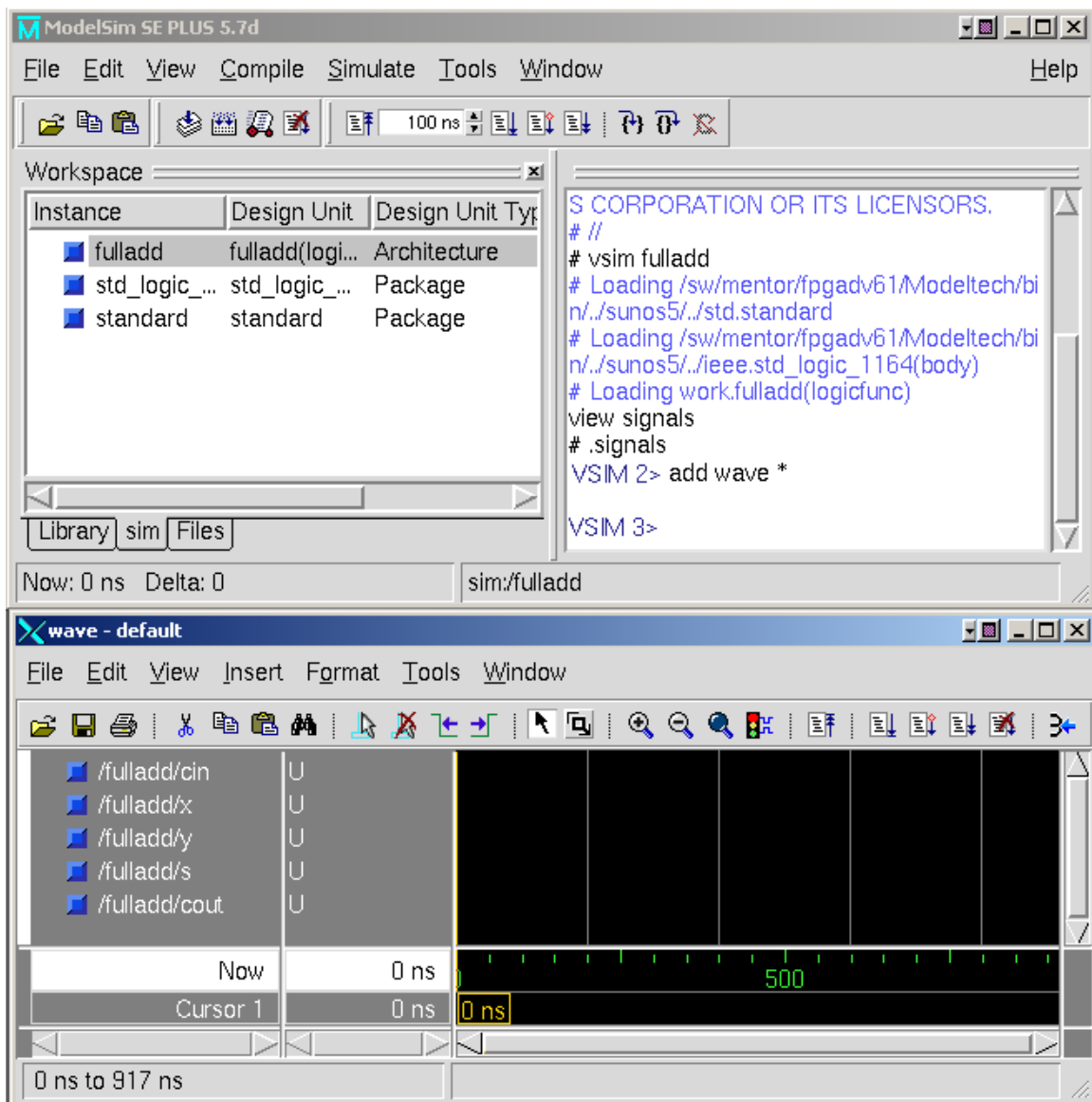
6. **Edit**→**Select All**

Note that the command you just selected was echoed in the *ModelSim* command window. In fact, any command exercised graphically can be typed directly or, as will be shown later, can be executed from a script file.

Next, return the mouse cursor to the *ModelSim* command window and type:

7. **add wave ***

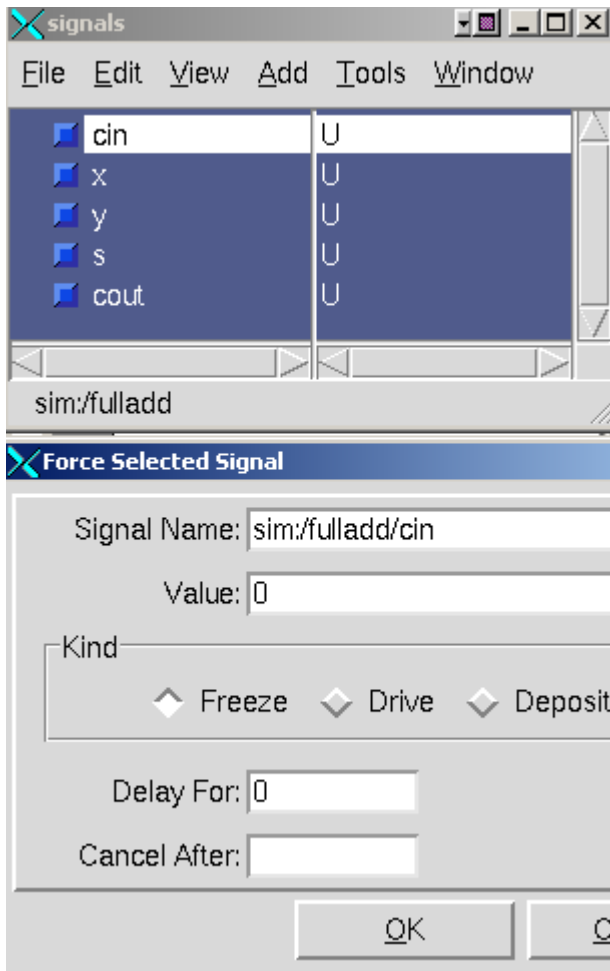
This will bring up a waveform window:



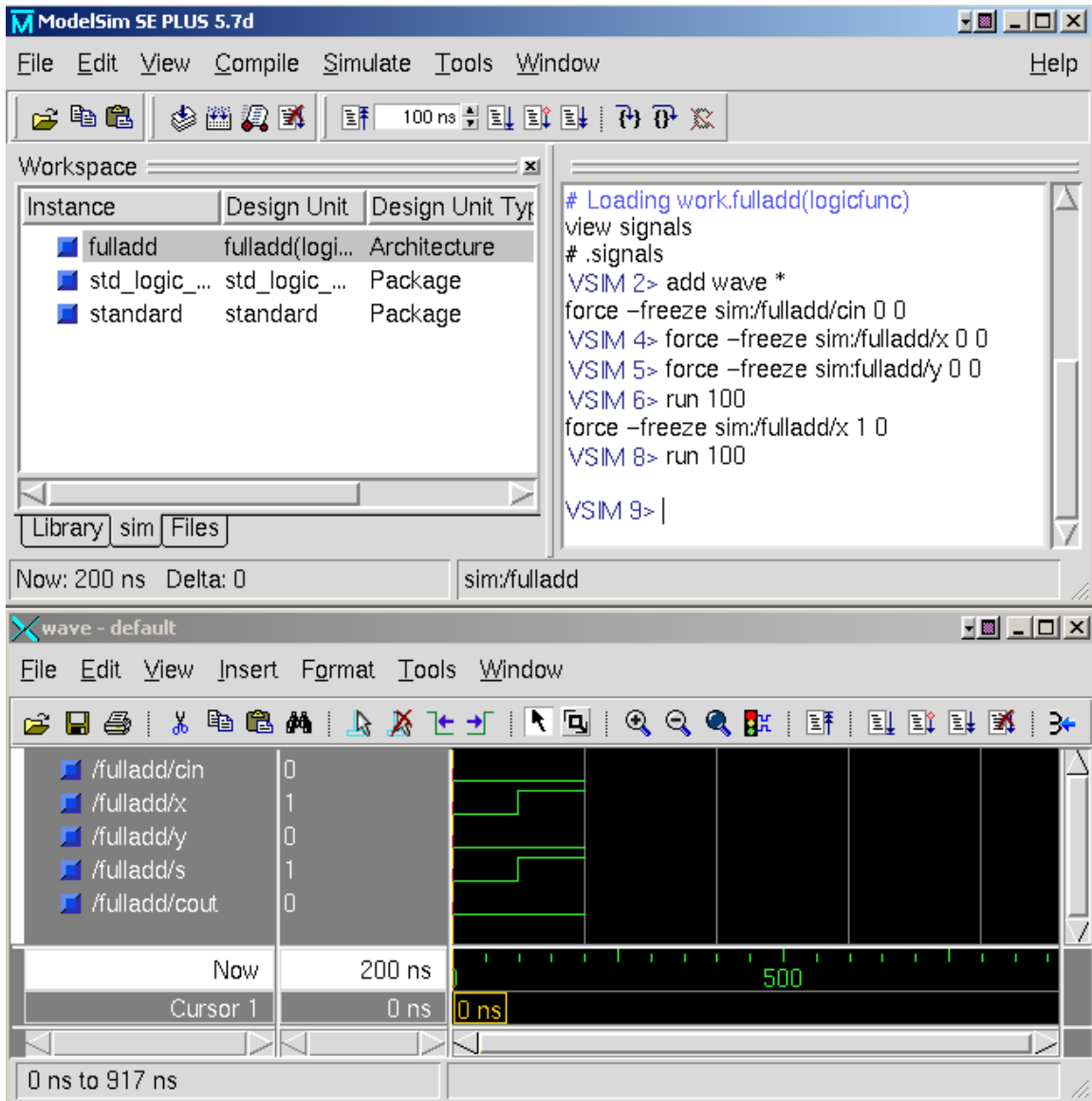
Next, return to the Signals window and select a signal of interest using the left-mouse-button. Then, from the pull-down menu, select:

8. **Edit → Force**

Fill in the dialog box as indicated in the new window and click OK:



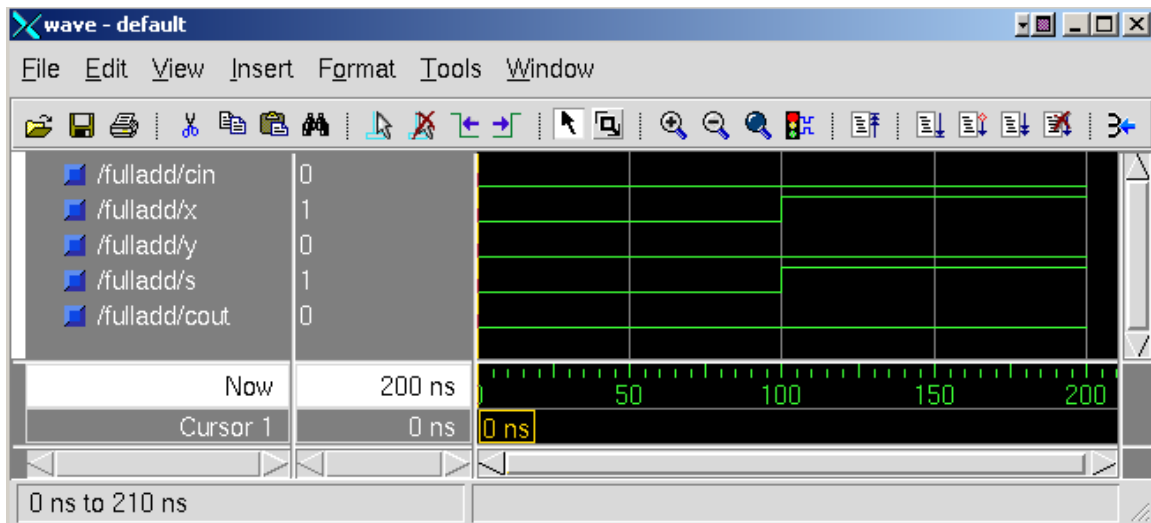
Alternatively, you can type in the *ModelSim* command window the values you want and then run the simulation. In this case, force all of the inputs to zero for the first 100 ns and then force *x* to be a 1 for the next 100 ns. Note the output signal *s* is changed:



The waveform window can be manipulated using the View options. For example, selecting:

9. **View→Zoom→Zoom Full**

This will bring up the following window:



You should continue forcing inputs and running the simulation until you are convinced that the design exhibits the desired behavior. When finished, type in the *ModelSim* command window:

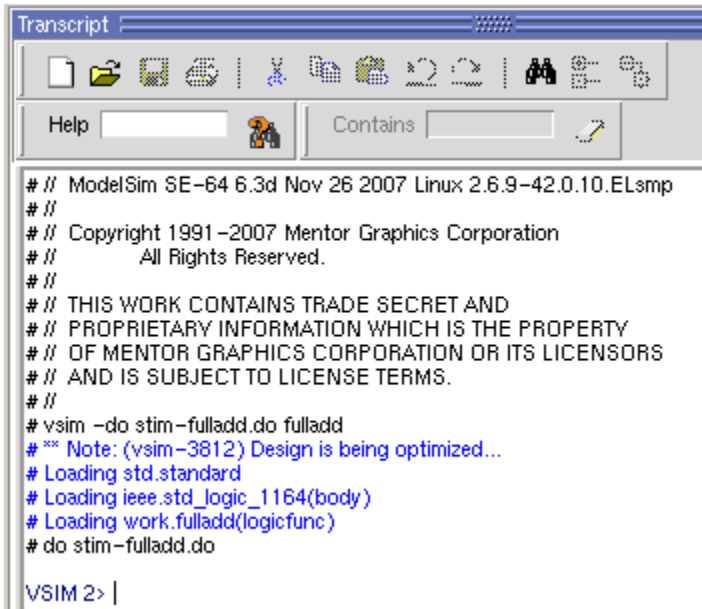
10. **quit -force**

If you encounter an error, you should edit the vhd source file and redo the steps above until you are certain your design is okay.

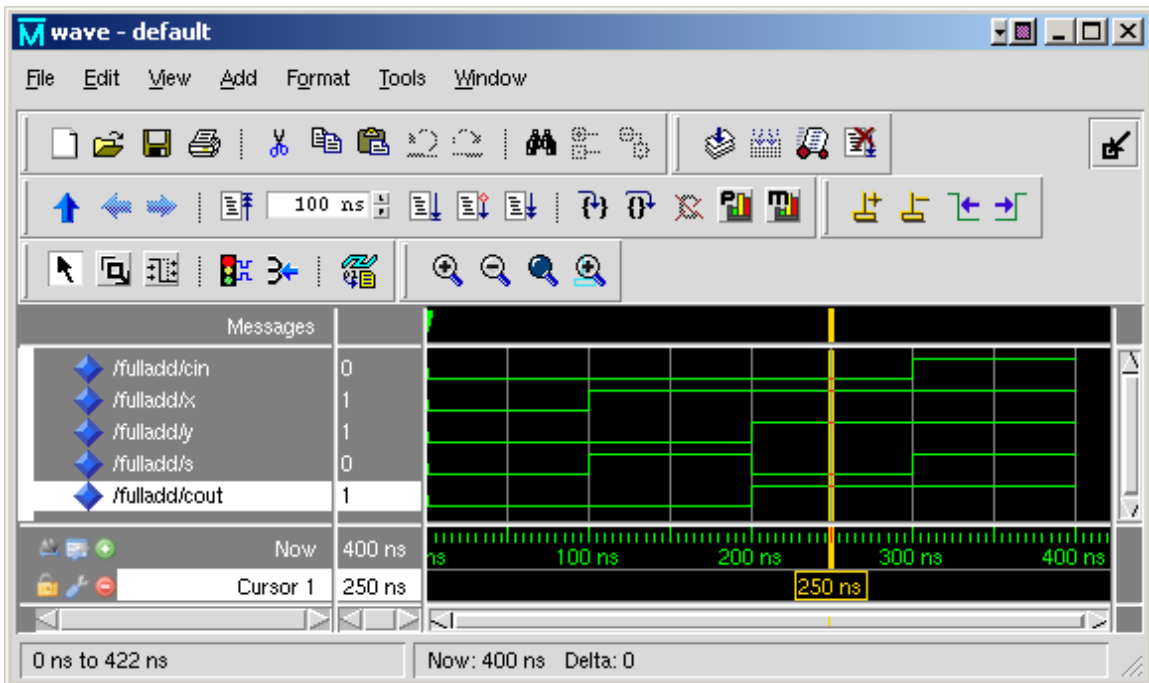
As stated previously, instead of selecting commands graphically using the menus or typing them individually into the command window, you can use a script file. The file “stim-fulladd.do” contains several inputs which will automatically be executed by typing:

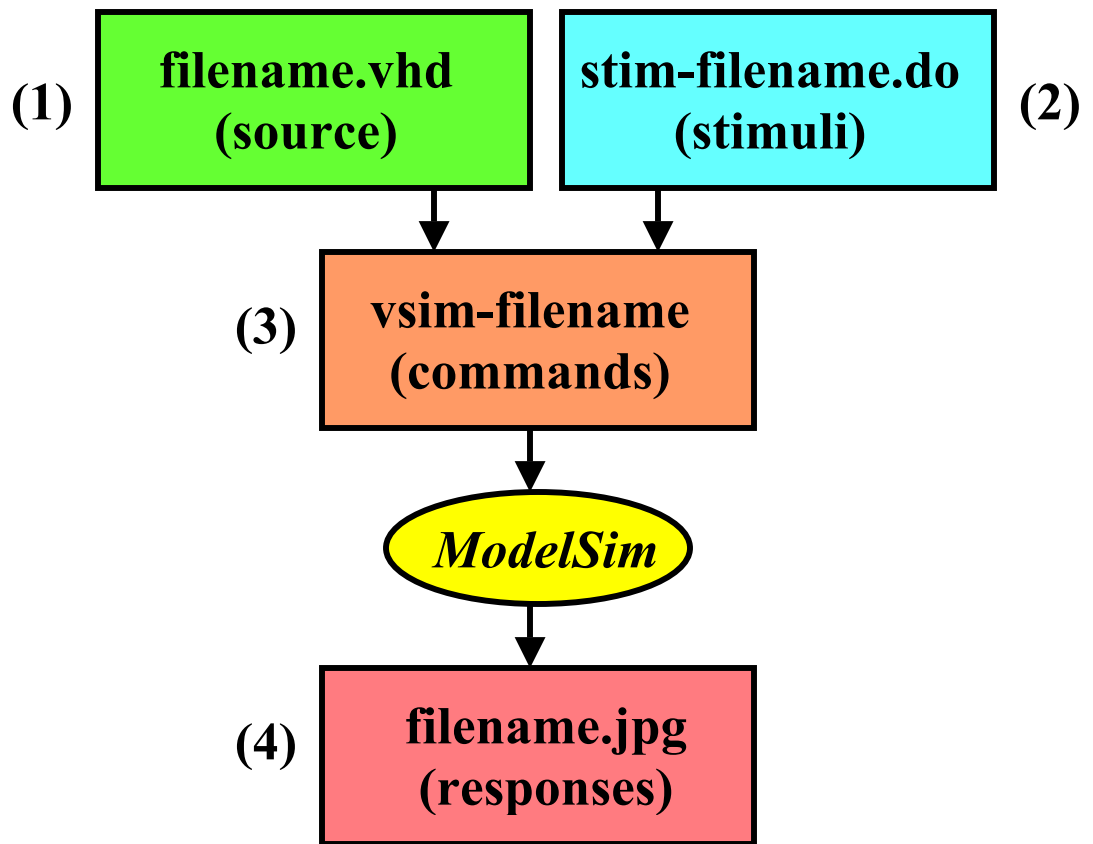
11. `./vsim-fulladd` (Note: Be sure this file is executable—`chmod u+x`)

This produces the windows below (Version 6.3d). Note the position of the cursor in the wave window at 250 ns and the corresponding values near the signal names:



```
## ModelSim SE-64 6.3d Nov 26 2007 Linux 2.6.9-42.0.10.ELsmp
##
## Copyright 1991-2007 Mentor Graphics Corporation
## All Rights Reserved.
##
## THIS WORK CONTAINS TRADE SECRET AND
## PROPRIETARY INFORMATION WHICH IS THE PROPERTY
## OF MENTOR GRAPHICS CORPORATION OR ITS LICENSORS
## AND IS SUBJECT TO LICENSE TERMS.
##
# vsim -do stim-fulladd.do fulladd
# ** Note: (vsim-3812) Design is being optimized...
# Loading std.standard
# Loading ieee.std_logic_1164(body)
# Loading work.fulladd(logicfunc)
# do stim-fulladd.do
VSIM 2> |
```





The hierarchical implementation of a 4-bit adder can be simulated by typing:

12. `./vsim-adder4`

This produces the windows below. Note the position of the cursor in the wave window at 250 ns and the corresponding values near the signal names:

The screenshot shows two windows from the ModelSim SE-64 environment. The top window is a command prompt with the following text:

```
## ModelSim SE-64 6.3d Nov 26 2007 Linux 2.6.9-42.0.10.ELsmp
##
## Copyright 1991-2007 Mentor Graphics Corporation
## All Rights Reserved.
##
## THIS WORK CONTAINS TRADE SECRET AND
## PROPRIETARY INFORMATION WHICH IS THE PROPERTY
## OF MENTOR GRAPHICS CORPORATION OR ITS LICENSORS
## AND IS SUBJECT TO LICENSE TERMS.
##
# vsim -do stim-adder4.do adder4
# ** Note: (vsim-3812) Design is being optimized...
# Loading std.standard
# Loading ieee.std_logic_1164(body)
# Loading work.adder4(structure)
# Loading work.fulladd(logicfunc)
# do stim-adder4.do
VSIM 2>
```

The bottom window is titled "wave - default" and displays a logic analyzer interface. On the left, a tree view shows the signal hierarchy for the adder4 component, including inputs (/adder4/cin), intermediate signals (/adder4/x, /adder4/y, /adder4/s), and outputs (/adder4/cout, /adder4/c). The main area shows a timing diagram with a grid. A yellow vertical cursor is positioned at 250 ns. The signal values at this time are:

Signal Name	Value at 250 ns
/adder4/cin	0
/adder4/x	0001
(3)	0
(2)	0
(1)	0
(0)	1
/adder4/y	0001
/adder4/s	0010
/adder4/cout	0
/adder4/c	100

The timing diagram also shows a scale bar at the bottom indicating 0 ns to 483 ns, with a current time of 400 ns and a delta of 0.

Part B – Testing Additional VHDL Files

Develop appropriate stimuli for each of these files and test them using ModelSim.

Capture the waveform produced for each one and link to your restricted webpage.

func3.vhd

mux2to1.vhd

dec2to4.vhd

seg7.vhd

upcount.vhd

BCDcount.vhd

simple.vhd

Just observe the following VHDL files. You do not have to simulate them.

onepulse.vhd

clk_div.vhd

debounce.vhd

hierarch.vhd

Appendix A -- Fulladd

fulladd.vhd

```
1  |-- Brown Example Appendix A.7a
2  LIBRARY ieee ;
3  USE ieee.std_logic_1164.all ;
4
5  ENTITY fulladd IS
6      PORT ( Cin, x, y : IN  STD_LOGIC ;
7            s, Cout : OUT  STD_LOGIC ) ;
8  END fulladd ;
9
10 ARCHITECTURE LogicFunc OF fulladd IS
11 BEGIN
12     s <= x XOR y XOR Cin ;
13     Cout <= (x AND y) OR (x AND Cin) OR (y AND Cin) ;
14 END LogicFunc ;
--
```

vsim-fulladd

```
vlib work
vcom fulladd.vhd
vsim fulladd -do stim-fulladd.do
```

stim-fulladd.do

```
add wave *
force -freeze sim:/fulladd/cin 0 0
force -freeze sim:/fulladd/x 0 0
force -freeze sim:/fulladd/y 0 0
run 100
force -freeze sim:/fulladd/cin 0 0
force -freeze sim:/fulladd/x 1 0
force -freeze sim:/fulladd/y 0 0
run 100
force -freeze sim:/fulladd/cin 0 0
force -freeze sim:/fulladd/x 1 0
force -freeze sim:/fulladd/y 1 0
run 100
force -freeze sim:/fulladd/cin 1 0
```

```
force -freeze sim:/fulladd/x 1 0
force -freeze sim:/fulladd/y 1 0
run 100
```

Appendix B – Adder4

adder4.vhd

```
1  -- Brown Example Appendix A.7b
2  LIBRARY ieee ;
3  USE ieee.std_logic_1164.all ;
4
5  ENTITY adder4 IS
6      PORT ( Cin : IN  STD_LOGIC ;
7            X, Y : IN  STD_LOGIC_VECTOR(3 DOWNTO 0) ;
8            S   : OUT  STD_LOGIC_VECTOR(3 DOWNTO 0) ;
9            Cout : OUT  STD_LOGIC ) ;
10 END adder4 ;
11
12 ARCHITECTURE Structure OF adder4 IS
13     SIGNAL C : STD_LOGIC_VECTOR(1 TO 3) ;
14     COMPONENT fulladd
15         PORT ( Cin, x, y : IN  STD_LOGIC ;
16              s, Cout : OUT  STD_LOGIC ) ;
17     END COMPONENT ;
18 BEGIN
19     stage0: fulladd PORT MAP ( Cin , X(0), Y(0), S(0), C(1) ) ;
20     stage1: fulladd PORT MAP ( C(1), X(1), Y(1), S(1), C(2) ) ;
21     stage2: fulladd PORT MAP ( C(2), X(2), Y(2), S(2), C(3) ) ;
22     stage3: fulladd PORT MAP (
23         x => X(3), y => Y(3), Cin => C(3), s => S(3), Cout => Cout ) ;
24 END Structure ;
```

vsim-adder4

```
vlib work
vcom -work work adder4.vhd
vsim adder4 -do stim-adder4.do
```

stim-adder4.do

```
add wave *
force -freeze sim:/adder4/cin 0 0
force -freeze sim:/adder4/x 0000 0
force -freeze sim:/adder4/y 0000 0
run 100
force -freeze sim:/adder4/cin 0 0
force -freeze sim:/adder4/x 0001 0
```

```
force -freeze sim:/adder4/y 0000 0
run 100
force -freeze sim:/adder4/cin 0 0
force -freeze sim:/adder4/x 0001 0
force -freeze sim:/adder4/y 0001 0
run 100
force -freeze sim:/adder4/cin 1 0
force -freeze sim:/adder4/x 0001 0
force -freeze sim:/adder4/y 0001 0
run 100
```