# Debugging your Project

Digital System Design

2019

Professor Jonathan H. Manton

# Introduction

- The workshops already taught you how to write a test bench and debug your code using ModelSim
  - This short note is to remind you what you learnt
- You can complete essentially the whole project (e.g., at home) without using the DE1-SoC board by using ModelSim
- Write your test bench first, then write the Verilog code
  - Repeat for each stage of the project

# Test Bench Demo

Create a New Verilog file in Quartus and save as testStage4.v

```
module testStage4;
  reg signed [7:0] x;
  reg enable;
  wire [6:0] H3, H2, H1, H0;

  Disp2cNum DUT(.x(x), .enable(enable), .H3(H3), .H2(H2), .H1(H1), .H0(H0));

    initial begin
      enable = 0;
      x = 0;
      #1
      enable = 1;
      #1
      x = 5;
      #1
      x = -5;
      #1
      x = 12;
      #1
      x = -12;
      #1
      x = 123;
      #1
      x = -123;
      #1
      $stop;
    end
  endmodule
```

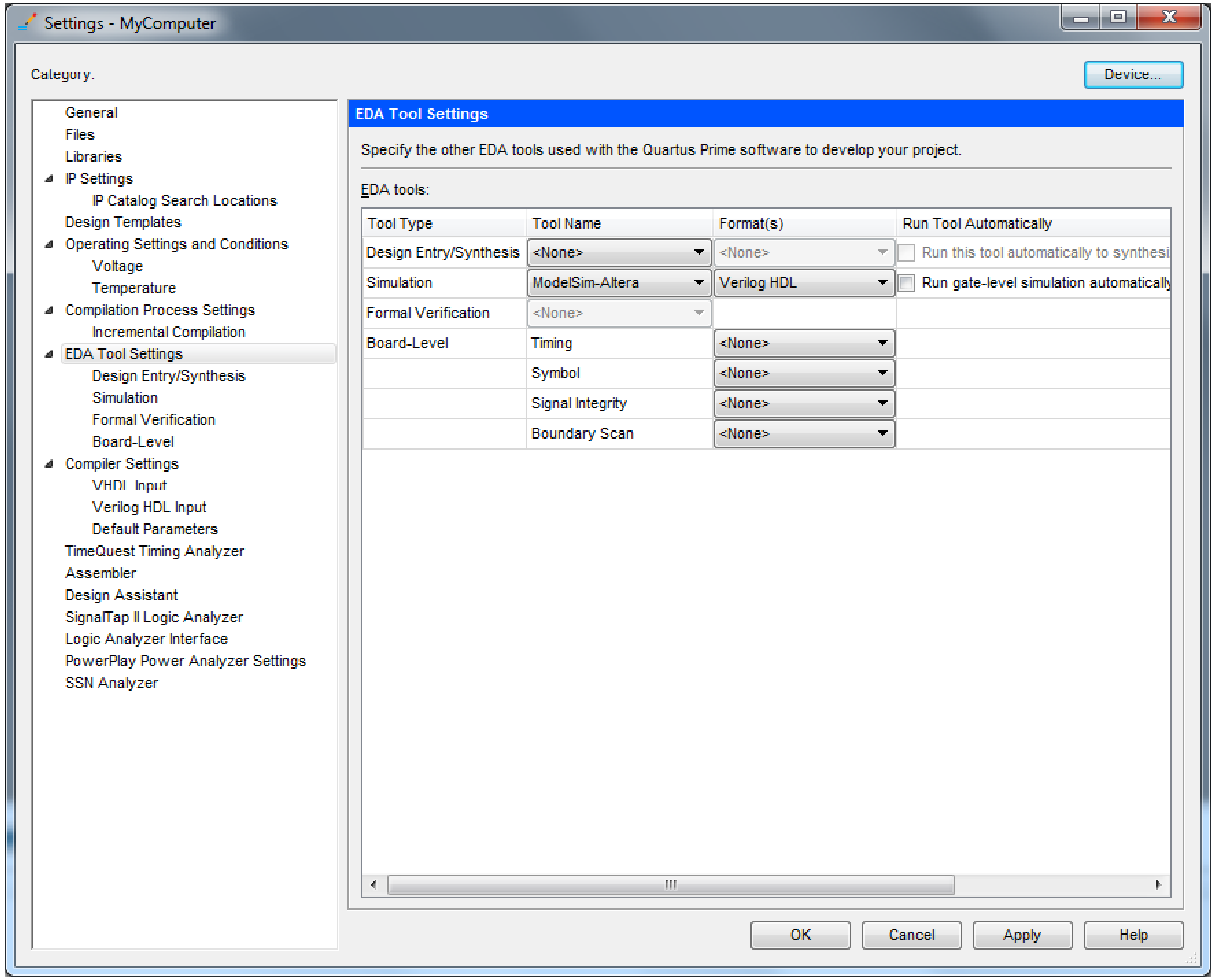
You write this module in Stage 4 of Project 1

How do we decide what values to use to test the Disp2cNum module?

Cannot end with a delay or ModelSim is unhappy, so add $stop;

A $finish; would cause ModelSim to exit!

- Select “ModelSim-Altera” for the simulation tool
  - And “Verilog HDL” as the format



Settings - MyComputer

Category:

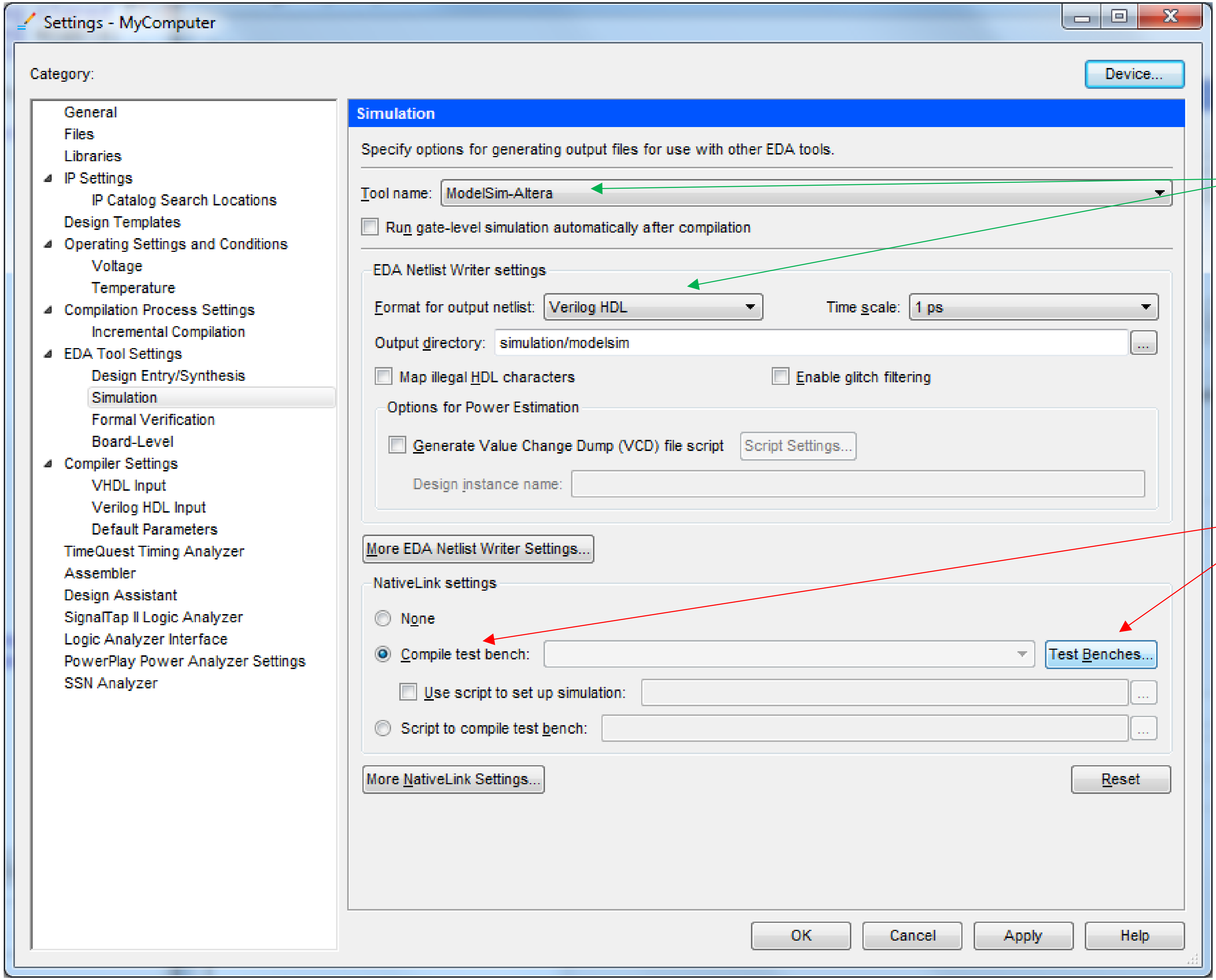
- General
- Files
- Libraries
- IP Settings
  - IP Catalog Search Locations
- Design Templates
- Operating Settings and Conditions
  - Voltage
  - Temperature
- Compilation Process Settings
  - Incremental Compilation
- EDA Tool Settings
  - Design Entry/Synthesis
  - Simulation
  - Formal Verification
  - Board-Level
- Compiler Settings
  - VHDL Input
  - Verilog HDL Input
  - Default Parameters
- TimeQuest Timing Analyzer
- Assembler
- Design Assistant
- SignalTap II Logic Analyzer
- Logic Analyzer Interface
- PowerPlay Power Analyzer Settings
- SSN Analyzer

EDA Tool Settings

Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

| Tool Type | Tool Name | Format(s) | Run Tool Automatically |
|---|---|---|---|
| Design Entry/Synthesis | <None> | <None> | □ Run this tool automatically to synthesi |
| Simulation | ModelSim-Altera | Verilog HDL | □ Run gate-level simulation automatically |
| Formal Verification | <None> | | |
| Board-Level | Timing | <None> | |
| | Symbol | <None> | |
| | Signal Integrity | <None> | |
| | Boundary Scan | <None> | |



Settings - MyComputer

Category:

- General
- Files
- Libraries
- IP Settings
  - IP Catalog Search Locations
- Design Templates
- Operating Settings and Conditions
  - Voltage
  - Temperature
- Compilation Process Settings
  - Incremental Compilation
- EDA Tool Settings
  - Design Entry/Synthesis
  - Simulation
  - Formal Verification
  - Board-Level
- Compiler Settings
  - VHDL Input
  - Verilog HDL Input
  - Default Parameters
- TimeQuest Timing Analyzer
- Assembler
- Design Assistant
- SignalTap II Logic Analyzer
- Logic Analyzer Interface
- PowerPlay Power Analyzer Settings
- SSN Analyzer

Simulation

Specify options for generating output files for use with other EDA tools.

Tool name: ModelSim-Altera

□ Run gate-level simulation automatically after compilation

EDA Netlist Writer settings

Format for output netlist: Verilog HDL    Time scale: 1 ps

Output directory: simulation/modelsim

□ Map illegal HDL characters    □ Enable glitch filtering

Options for Power Estimation

□ Generate Value Change Dump (VCD) file script

Design instance name:

NativeLink settings

○ None

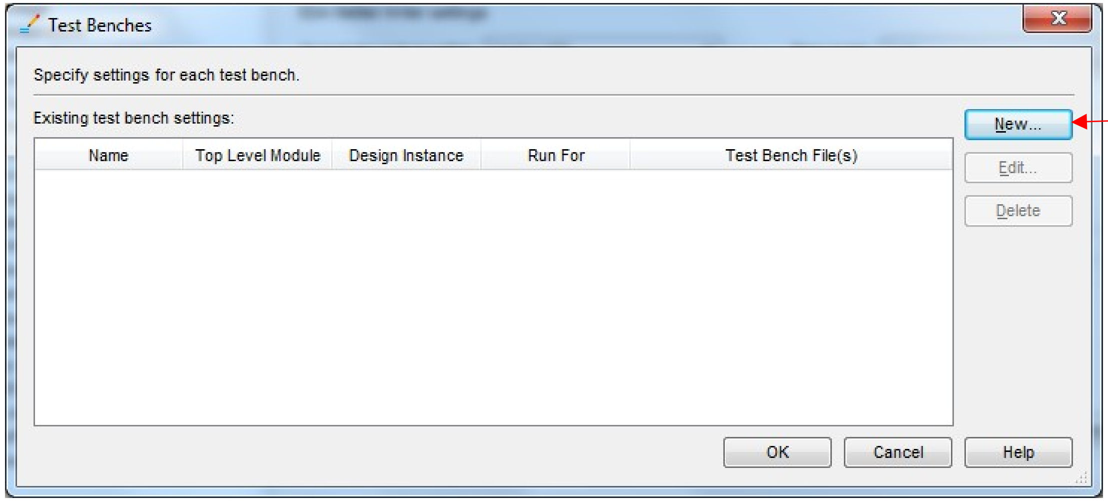
● Compile test bench:

□ Use script to set up simulation:

○ Script to compile test bench:

Alternatively, could have selected ModelSim-Altera and Verilog here

Click on Test Benches...



Specify settings for each test bench.

Existing test bench settings:

| Name | Top Level Module | Design Instance | Run For | Test Bench File(s) |
|---|---|---|---|---|

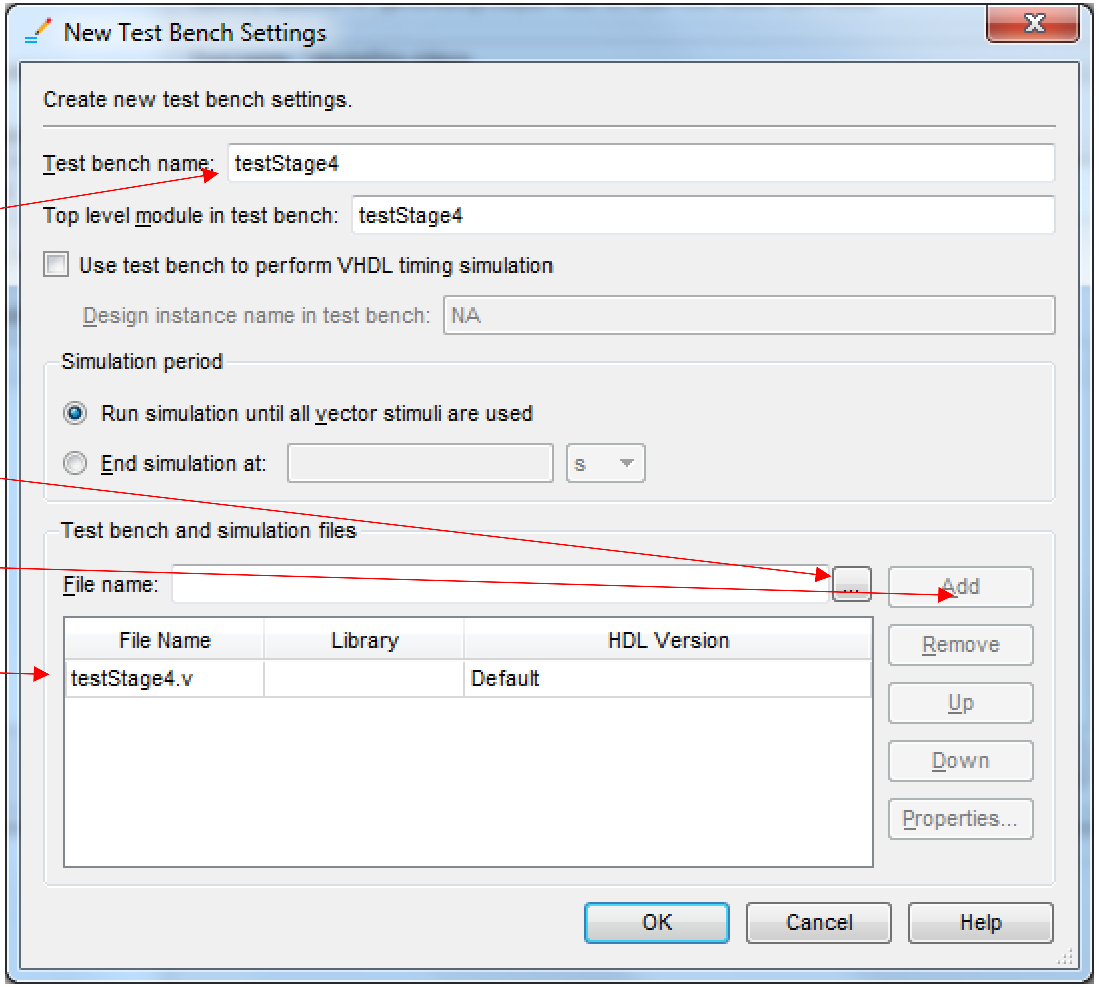
Click on New…

Enter Test bench name

Click on … then select testStage4.v

Click on Add to add it

End result is this



Create new test bench settings.

Test bench name: testStage4

Top level module in test bench: testStage4

□ Use test bench to perform VHDL timing simulation

Design instance name in test bench: NA

Simulation period

● Run simulation until all vector stimuli are used

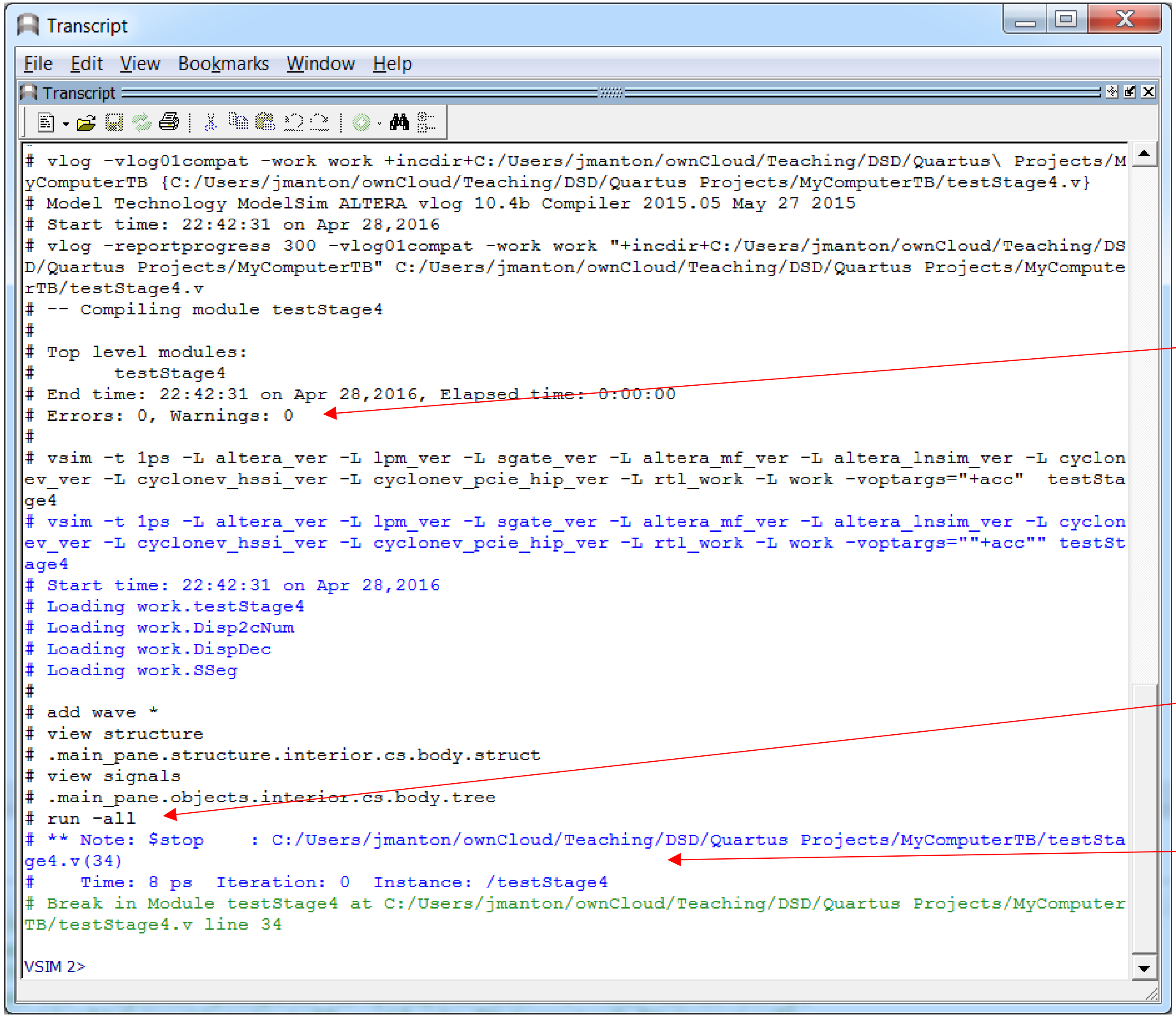
○ End simulation at: s

Test bench and simulation files

File name:

| File Name | Library | HDL Version |
|---|---|---|
| testStage4.v | | Default |

In Quartus, select Tools > Run Simulation Tool > RTL Simulation



```
# vlog -vlog01compat -work work +incdir+C:/Users/jmanton/ownCloud/Teaching/DSD/Quartus\ Projects/M
yComputerTB {C:/Users/jmanton/ownCloud/Teaching/DSD/Quartus Projects/MyComputerTB/testStage4.v}
# Model Technology ModelSim ALTERA vlog 10.4b Compiler 2015.05 May 27 2015
# Start time: 22:42:31 on Apr 28,2016
# vlog -reportprogress 300 -vlog01compat -work work "+incdir+C:/Users/jmanton/ownCloud/Teaching/DS
D/Quartus Projects/MyComputerTB" C:/Users/jmanton/ownCloud/Teaching/DSD/Quartus Projects/MyCompute
rTB/testStage4.v
# -- Compiling module testStage4
#
# Top level modules:
#       testStage4
# End time: 22:42:31 on Apr 28,2016, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
#
# vsim -t 1ps -L altera_ver -L lpm_ver -L sgate_ver -L altera_mf_ver -L altera_lnsim_ver -L cyclon
ev_ver -L cyclonev_hssi_ver -L cyclonev_pcie_hip_ver -L rtl_work -L work -voptargs="+acc"  testSta
ge4
# vsim -t 1ps -L altera_ver -L lpm_ver -L sgate_ver -L altera_mf_ver -L altera_lnsim_ver -L cyclon
ev_ver -L cyclonev_hssi_ver -L cyclonev_pcie_hip_ver -L rtl_work -L work -voptargs=""+acc"" testSt
age4
# Start time: 22:42:31 on Apr 28,2016
# Loading work.testStage4
# Loading work.Disp2cNum
# Loading work.DispDec
# Loading work.SSeg
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: $stop    : C:/Users/jmanton/ownCloud/Teaching/DSD/Quartus Projects/MyComputerTB/testSta
ge4.v(34)
#    Time: 8 ps  Iteration: 0  Instance: /testStage4
# Break in Module testStage4 at C:/Users/jmanton/ownCloud/Teaching/DSD/Quartus Projects/MyComputer
TB/testStage4.v line 34

VSIM 2>
```

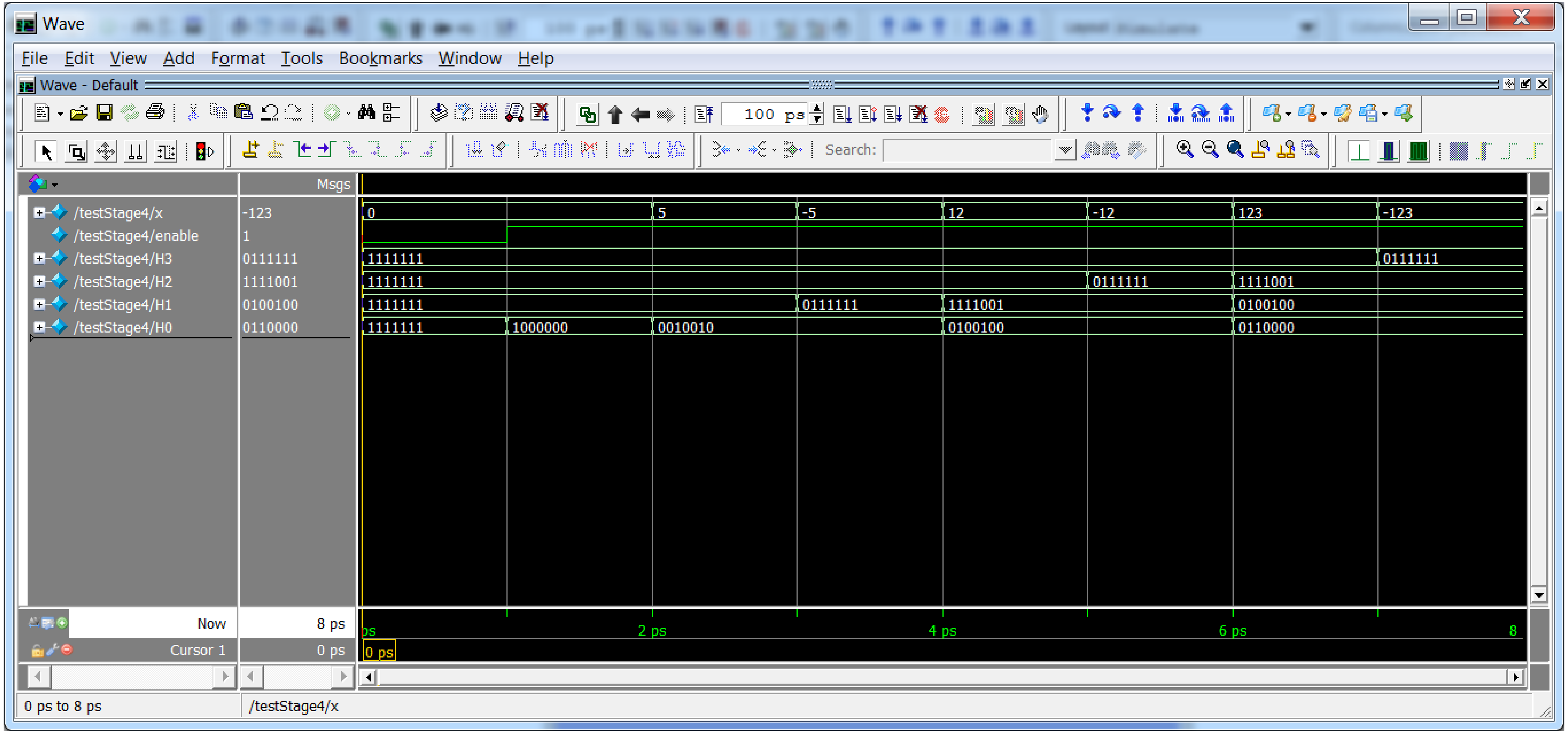
Quartus calls ModelSim for you

Check that compilation is successful

Automatically running simulation

Reached $stop statement

The waves have already been added to the Wave window for us. (Need to “zoom full” and adjust the grid units if you wish to recreate this figure exactly.)



| Signal | Msgs |
|---|---|
| /testStage4/x | -123 |
| /testStage4/enable | 1 |
| /testStage4/H3 | 0111111 |
| /testStage4/H2 | 1111001 |
| /testStage4/H1 | 0100100 |
| /testStage4/H0 | 0110000 |

x: 0, 5, -5, 12, -12, 123, -123

H3: 1111111, 0111111

H2: 1111111, 0111111, 1111001

H1: 1111111, 0111111, 1111001, 0100100

H0: 1111111, 1000000, 0010010, 0100100, 0110000

Now 8 ps

Cursor 1 0 ps

0 ps

0 ps to 8 ps

/testStage4/x

How can we check if our module is working correctly?

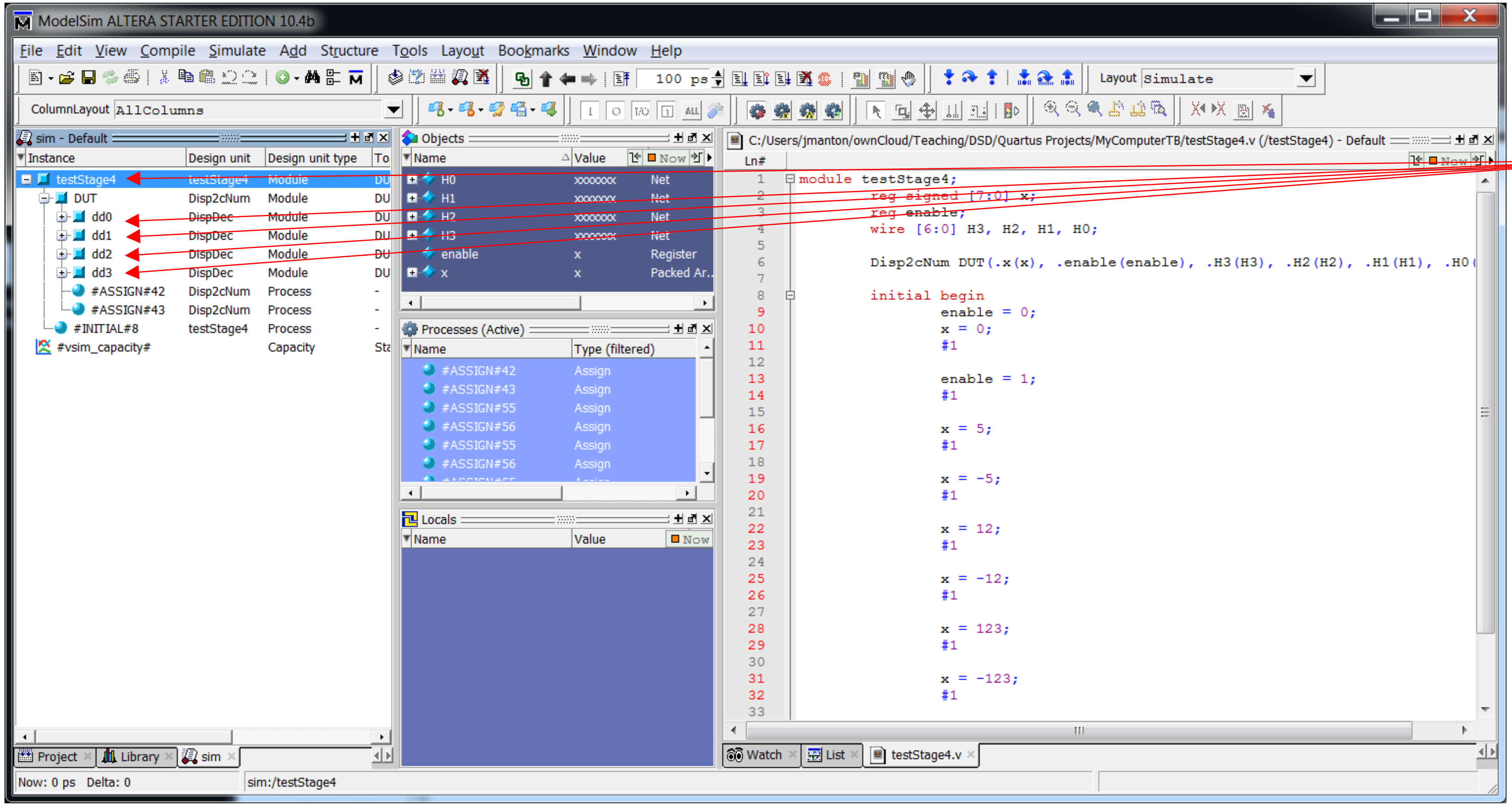
# Options for Checking

1. Look up the correct 7-segment values and manually check
2. Look up the correct 7-segment values and automatically check
3. Write a function or module that decodes the 7-segment value, then check the decoded digits are correct
4. Ask ModelSim to display the signals being passed to the 7-segment encoder module

End Simulation (because we cannot add new waveforms and see the result without going back to the start)
[ Or use Restart…, or we could have had a test bench that runs forever. ]

From the Library pane, double-click on work > testStage 4

Expand DUT, and see dd0, dd1, dd2, dd3



| Instance | Design unit | Design unit type | To |
|---|---|---|---|
| testStage4 | testStage4 | Module | DU |
| DUT | Disp2cNum | Module | DU |
| dd0 | DispDec | Module | DU |
| dd1 | DispDec | Module | DU |
| dd2 | DispDec | Module | DU |
| dd3 | DispDec | Module | DU |
| #ASSIGN#42 | Disp2cNum | Process | - |
| #ASSIGN#43 | Disp2cNum | Process | - |
| #INITIAL#8 | testStage4 | Process | - |
| #vsim_capacity# |  | Capacity | Sta |

| Name | Value | Now |
|---|---|---|
| H0 | xxxxxxx | Net |
| H1 | xxxxxxx | Net |
| H2 | xxxxxxx | Net |
| H3 | xxxxxxx | Net |
| enable | x | Register |
| x | x | Packed Ar.. |

| Name | Type (filtered) |
|---|---|
| #ASSIGN#42 | Assign |
| #ASSIGN#43 | Assign |
| #ASSIGN#55 | Assign |
| #ASSIGN#56 | Assign |
| #ASSIGN#55 | Assign |
| #ASSIGN#56 | Assign |

| Name | Value | Now |
|---|---|---|

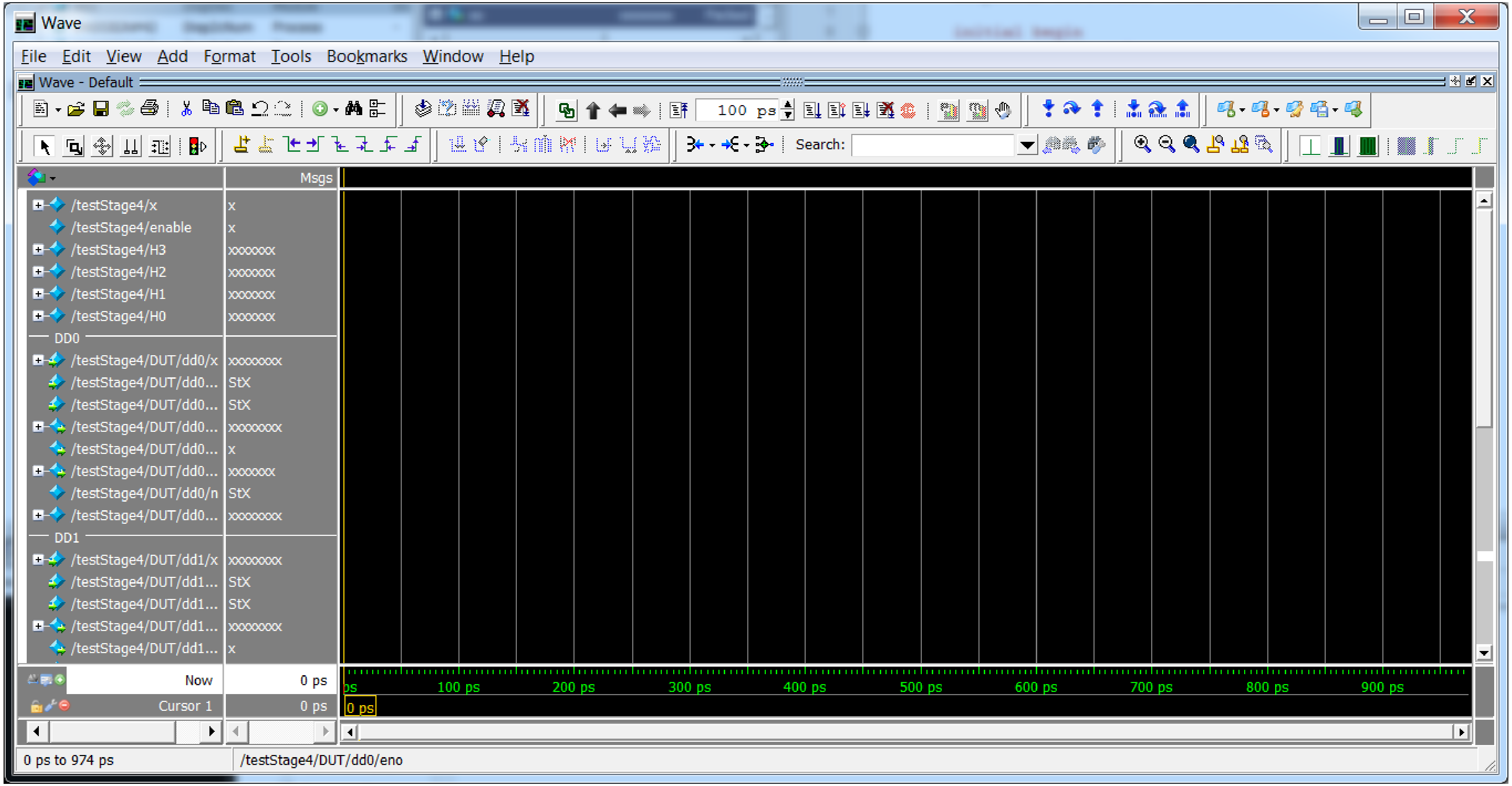
C:/Users/jmanton/ownCloud/Teaching/DSD/Quartus Projects/MyComputerTB/testStage4.v (/testStage4) - Default

```
module testStage4;
        reg signed [7:0] x;
        reg enable;
        wire [6:0] H3, H2, H1, H0;

        Disp2cNum DUT(.x(x), .enable(enable), .H3(H3), .H2(H2), .H1(H1), .H0(

        initial begin
                enable = 0;
                x = 0;
                #1

                enable = 1;
                #1

                x = 5;
                #1

                x = -5;
                #1

                x = 12;
                #1

                x = -12;
                #1

                x = 123;
                #1

                x = -123;
                #1
```

Now: 0 ps Delta: 0 sim:/testStage4

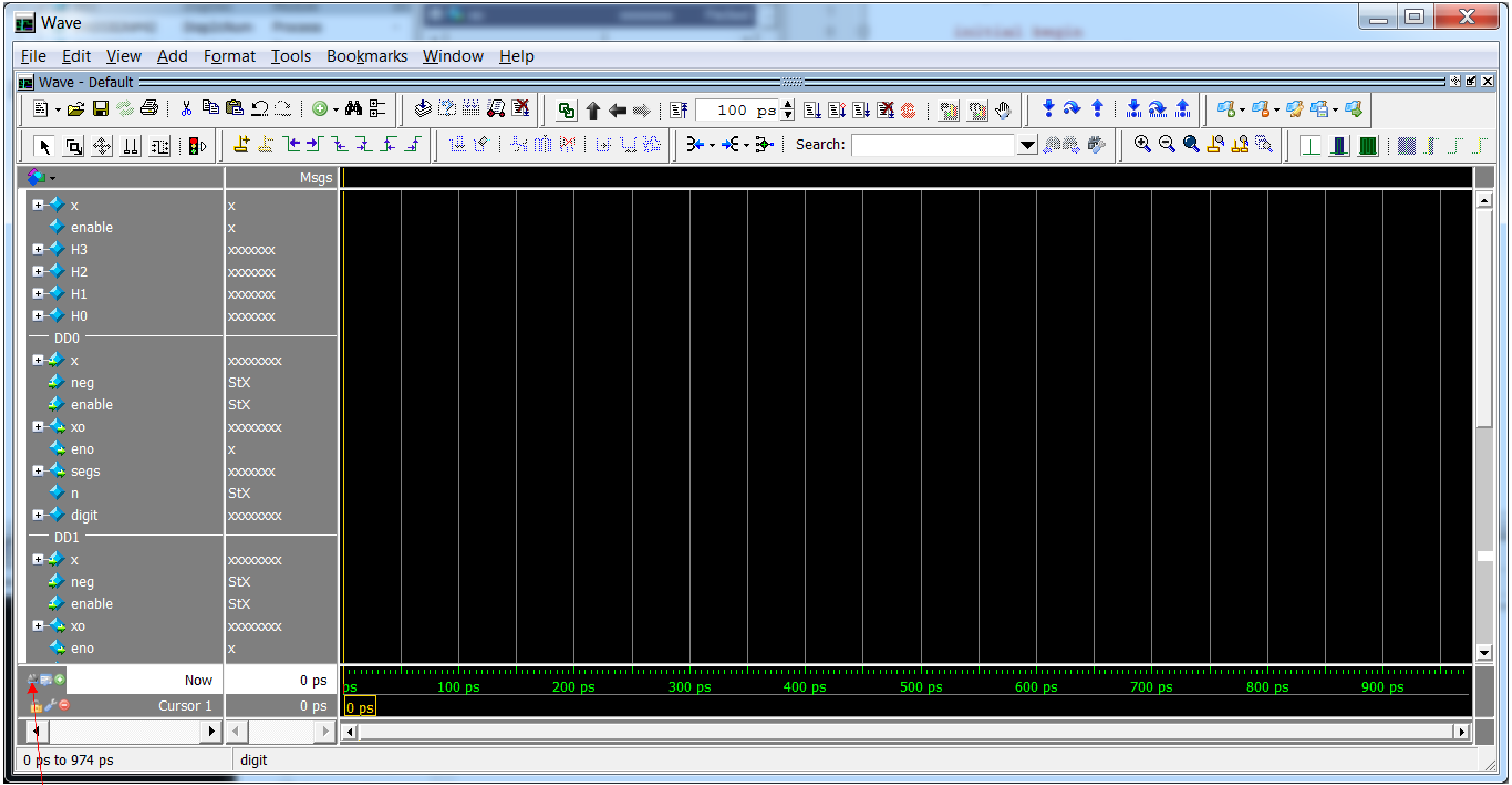
Right-click and select Add Wave

In the Wave pane, you can add Dividers to make things look a bit nicer…



| | Msgs |
|---|---|
| /testStage4/x | x |
| /testStage4/enable | x |
| /testStage4/H3 | xxxxxxx |
| /testStage4/H2 | xxxxxxx |
| /testStage4/H1 | xxxxxxx |
| /testStage4/H0 | xxxxxxx |
| DD0 | |
| /testStage4/DUT/dd0/x | xxxxxxxx |
| /testStage4/DUT/dd0... | StX |
| /testStage4/DUT/dd0... | StX |
| /testStage4/DUT/dd0... | xxxxxxxx |
| /testStage4/DUT/dd0... | x |
| /testStage4/DUT/dd0... | xxxxxxx |
| /testStage4/DUT/dd0/n | StX |
| /testStage4/DUT/dd0... | xxxxxxxx |
| DD1 | |
| /testStage4/DUT/dd1/x | xxxxxxxx |
| /testStage4/DUT/dd1... | StX |
| /testStage4/DUT/dd1... | StX |
| /testStage4/DUT/dd1... | xxxxxxxx |
| /testStage4/DUT/dd1... | x |
| Now | 0 ps |
| Cursor 1 | 0 ps |

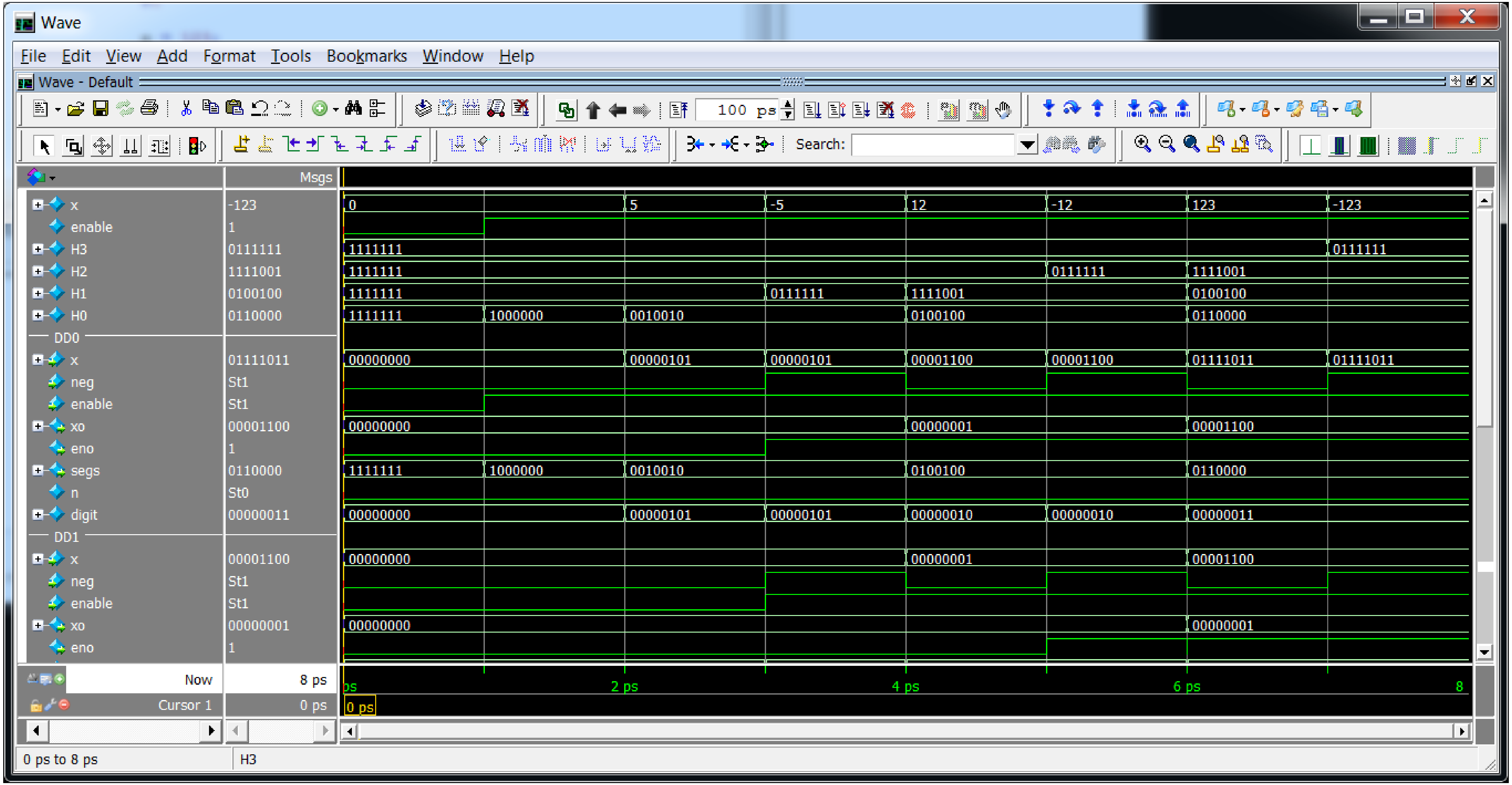
0 ps to 974 ps    /testStage4/DUT/dd0/eno



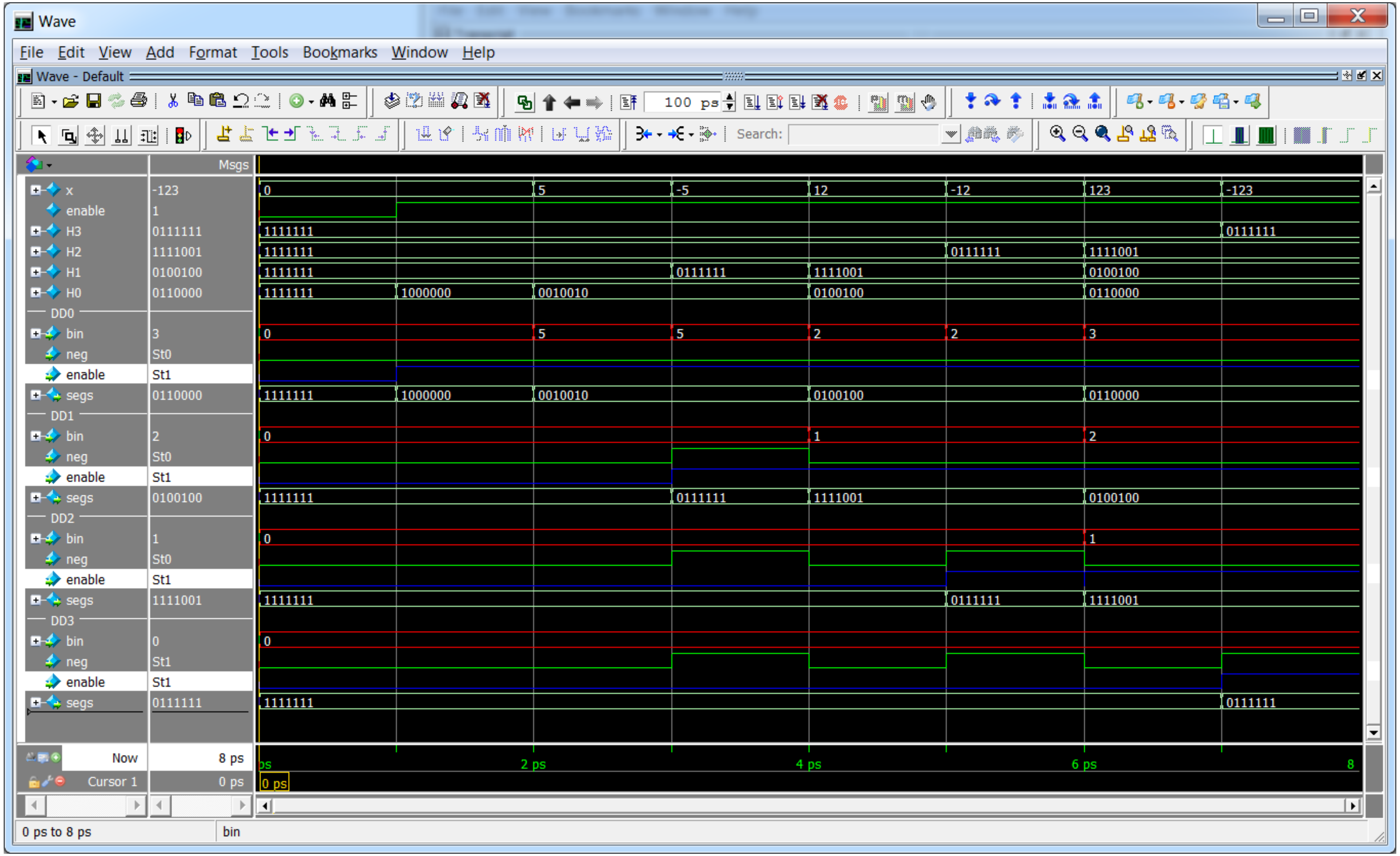
| | Msgs |
|---|---|
| x | x |
| enable | x |
| H3 | xxxxxxx |
| H2 | xxxxxxx |
| H1 | xxxxxxx |
| H0 | xxxxxxx |
| DD0 | |
| x | xxxxxxxx |
| neg | StX |
| enable | StX |
| xo | xxxxxxxx |
| eno | x |
| segs | xxxxxxx |
| n | StX |
| digit | xxxxxxxx |
| DD1 | |
| x | xxxxxxxx |
| neg | StX |
| enable | StX |
| xo | xxxxxxxx |
| eno | x |
| Now | 0 ps |
| Cursor 1 | 0 ps |

And then click here to get rid of the “path” in front of the variable names (if you think it looks nicer)

Simply click on “Run –All” (then click on Zoom Full). Might be useful for debugging!



Now go back (end simulation; double-click on work > testStage 4) and this time, right-click on the converter module under each dd module and select Add Wave, then Run –All.



| | Msgs |
|---|---|
| x | -123 |
| enable | 1 |
| H3 | 0111111 |
| H2 | 1111001 |
| H1 | 0100100 |
| H0 | 0110000 |
| DD0 | |
| bin | 3 |
| neg | St0 |
| enable | St1 |
| segs | 0110000 |
| DD1 | |
| bin | 2 |
| neg | St0 |
| enable | St1 |
| segs | 0100100 |
| DD2 | |
| bin | 1 |
| neg | St0 |
| enable | St1 |
| segs | 1111001 |
| DD3 | |
| bin | 0 |
| neg | St1 |
| enable | St1 |
| segs | 0111111 |
| Now | 8 ps |
| Cursor 1 | 0 ps |

0 ps to 8 ps bin

Change radix to decimal for "bin"

Change colour of certain waveforms

Check "bin", "neg" and "enable" are correct

# How to Debug?

- Ultimately, like driving a car
  - Practice!
- Add every relevant variable then look at waveforms
  - RTL can help tell you relevant variables
    - Filter on source and destination
- Use “watch”, “breakpoint”, “single-step” features of ModelSim
  - Allows you to go line by line through the Verilog code
- (Debugging is a generic skill; if you are good at debugging C code then you are likely going to be good at debugging Verilog code.)