VHDL – Lab 7

The master lab

Revision history

Date	Author	Description
2023-05-02	Kent Abrahamsson	First edition

1 Table of Contents

1 TABLE OF CONTENTS 3
2 GENERAL 4
2.1 Background 4
2.2 Intended outcome 4
2.3 Checkpoints 4
3 LAB INSTRUCTIONS 5
3.1 General 5
3.2 Running a testbench in Modelsim 5

2 General

2.1 Background

All previous lab assignments have been created to, step by step, introduce new challenges in VHDL design. In this assignment you will need to use all your skills and learn how a real project can be developed.

2.2 Intended outcome

In this lab the goal is to understand how a larger VHDL design can be specified and developed. The important concept of a well written testbench is again demonstrated.

2.3 Checkpoints

In order to Pass the lab assignment the student shall be able to show that the following tasks have been fulfilled.

- Properly written VHDL implementation. Use proper indentation, spaces are cheap! Add comments when you feel it's suitable.
- The design shall be implemented according to the Design specification.
- The design shall pass all testbench tests.
- The design shall be properly pin-planned and pass timing analysis.
- The worst case timing path for setup-requirement shall be checked and identified in code.
- The design shall pass all synthesis and Place & Route steps without any critical or strange warnings that can be solved easily.
- Design shall also be tested in hardware preferably with control from both key push buttons and Serial connection.

3 Lab instructions

3.1 General

The mission is to implement a PWM function according to a design specification. The design specification specifies how the design shall work and how the design shall be tested using a testbench.

You will be able to download a few design files from the course homepage, such as the Serial UART component, the PLL component, the reset control component, and help for handling a generic controlled instantiation in the top level.

The testbench that is designed to test the design according to its specification, and a constraints file that shall be used to constrain the clock frequency for timing analysis are also available in the course homepage.

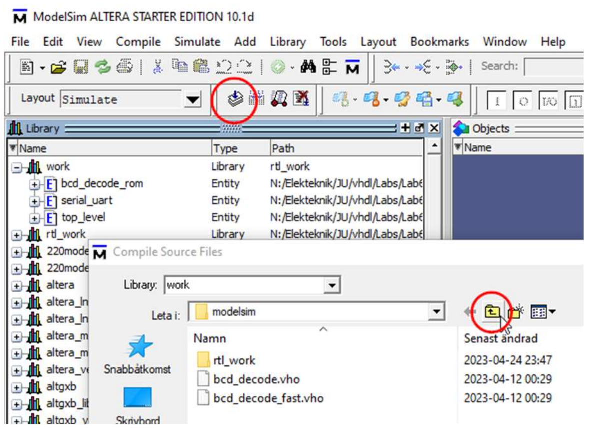
The idea is that when you have designed your design in a way that it completes the testbench it should also work in HW after a correct pin-planning is performed.

Since this assignment is more complex than other labs the suggestion is to team up in groups of **maximum 2 persons** and divide a few components between each other. The specification should be clear enough to understand what each sub component shall do. Ensure you read the design specification and understand the purpose of each sub-component in the design and also ensure that you understand the interface between the sub-components before starting.

3.2 Running a testbench in Modelsim

To run a testbench in Modelsim use the compile button and compile the testbench.

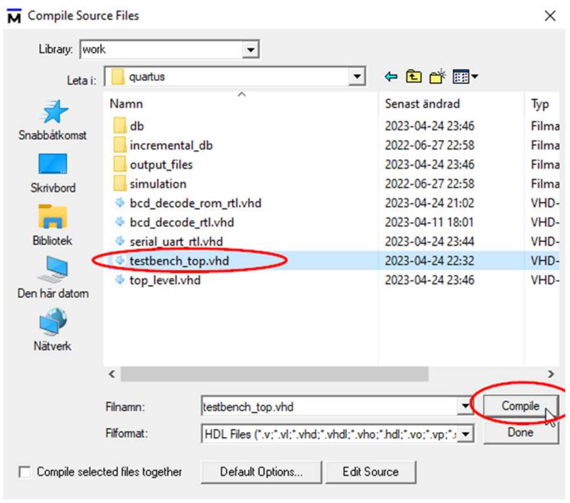
The testbench file is hopefully stored in the project folder so you will need to browse to it from the modelsim working directory, by clicking "up a level" in the folder structure.


ModelSim ALTERA STARTER EDITION 10.1d
File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
Search:
Layout Simulate
Library
Objects
Compile Source Files
Snabbåtkomst
Skrivbord

Name	Type	Path
work	Library	rtl_work
bcd_decode_rom	Entity	N:/Elekteknik/JU/vhdl/Labs/Lab6
serial_uart	Entity	N:/Elekteknik/JU/vhdl/Labs/Lab6
top_level	Entity	N:/Elekteknik/JU/vhdl/Labs/Lab6
rtl_work	Library	N:/Elekteknik/JU/vhdl/Labs/Lab6

Library: work
Leta i: modelsim

Namn	Senast ändrad
rtl_work	2023-04-24 23:47
bcd_decode.vho	2023-04-12 00:29
bcd_decode_fast.vho	2023-04-12 00:29

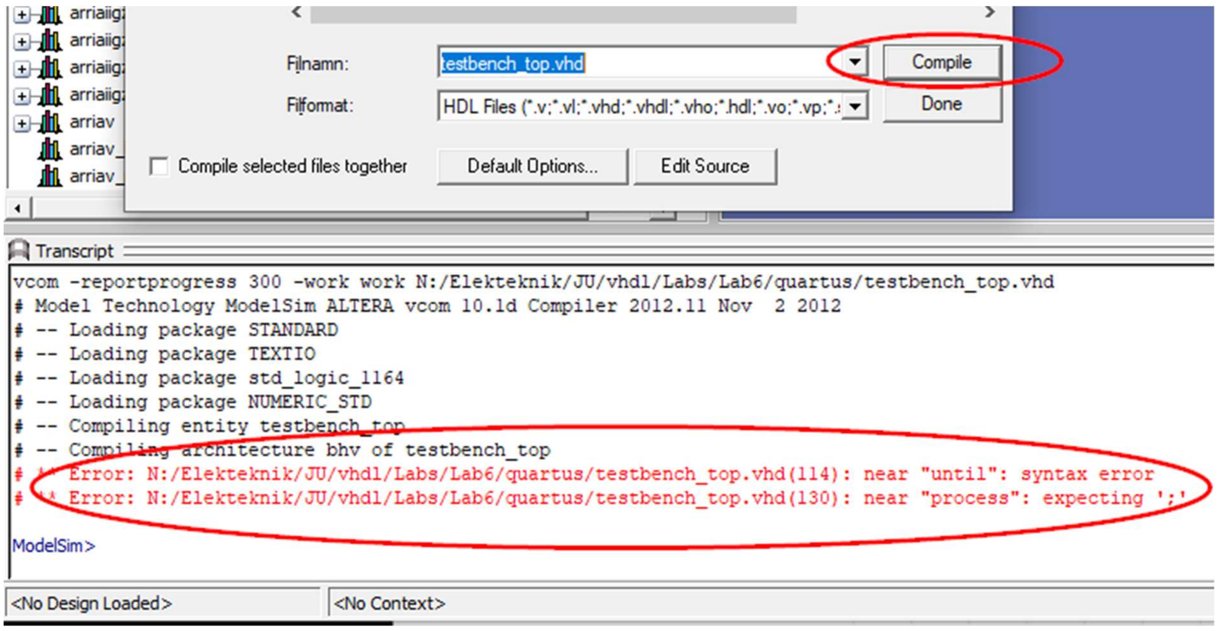


Library: work
Leta i: quartus

Namn	Senast ändrad	Typ
db	2023-04-24 23:46	Filma
incremental_db	2022-06-27 22:58	Filma
output_files	2023-04-24 23:46	Filma
simulation	2022-06-27 22:58	Filma
bcd_decode_rom_rtl.vhd	2023-04-24 21:02	VHD-
bcd_decode_rtl.vhd	2023-04-11 18:01	VHD-
serial_uart_rtl.vhd	2023-04-24 23:44	VHD-
testbench_top.vhd	2023-04-24 22:32	VHD-
top_level.vhd	2023-04-24 23:46	VHD-

Filnamn: testbench_top.vhd
Filformat: HDL Files (*.v;*.vl;*.vhd;*.vhdl;*.vho;*.hdl;*.vo;*.vp;*.

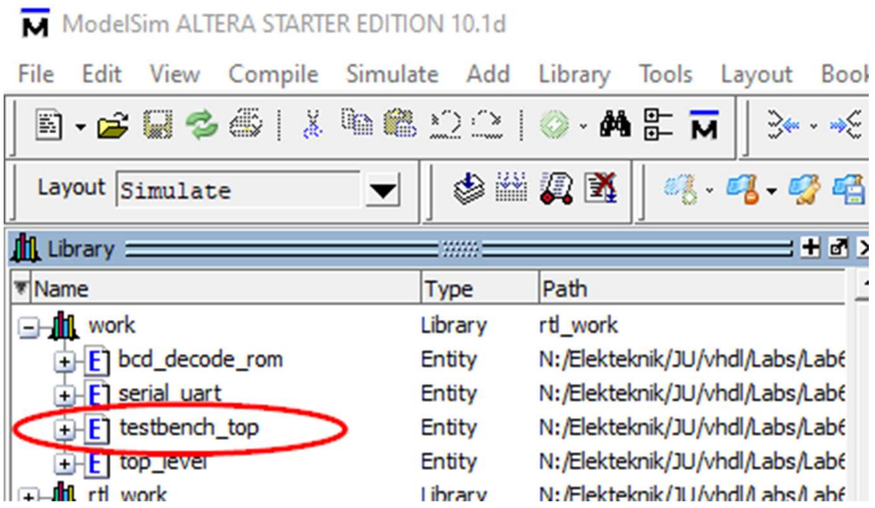
Check Modelsim Transcript window to ensure that no compile errors is found.


Filnamn: testbench_top.vhd
Filformat: HDL Files (*.v;*.vl;*.vhd;*.vhdl;*.vho;*.hdl;*.vo;*.vp;*.
Compile
Done
Compile selected files together
Default Options...
Edit Source
Transcript
<No Design Loaded>
<No Context>

```
vcom -reportprogress 300 -work work N:/Elekteknik/JU/vhdl/Labs/Lab6/quartus/testbench_top.vhd
# Model Technology ModelSim ALTERA vcom 10.1d Compiler 2012.11 Nov  2 2012
# -- Loading package STANDARD
# -- Loading package TEXTIO
# -- Loading package std_logic_1164
# -- Loading package NUMERIC_STD
# -- Compiling entity testbench_top
# -- Compiling architecture bhv of testbench_top
# ** Error: N:/Elekteknik/JU/vhdl/Labs/Lab6/quartus/testbench_top.vhd(114): near "until": syntax error
# ** Error: N:/Elekteknik/JU/vhdl/Labs/Lab6/quartus/testbench_top.vhd(130): near "process": expecting ';'

ModelSim>
```

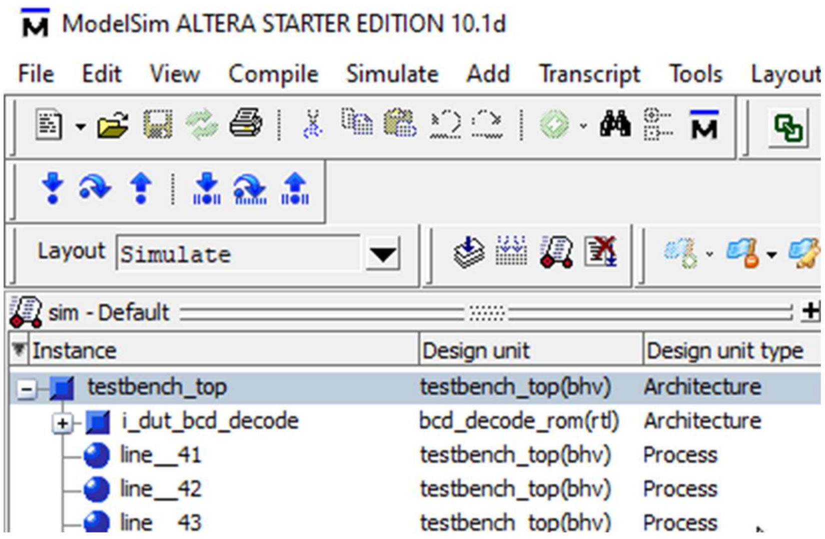
Once the testbench is compiled successfully you should be able to find it in the library “work”.



Name	Type	Path
work	Library	rtl_work
bcd_decode_rom	Entity	N:/Elekteknik/JU/vhdl/Labs/Lab6
serial_uart	Entity	N:/Elekteknik/JU/vhdl/Labs/Lab6
testbench_top	Entity	N:/Elekteknik/JU/vhdl/Labs/Lab6
top_level	Entity	N:/Elekteknik/JU/vhdl/Labs/Lab6
rtl_work	Library	N:/Elekteknik/JU/vhdl/Labs/Lab6

Double click the testbench top entity to start simulation.

Simulation should start and the DUT should be shown as an instance.


ModelSim ALTERA STARTER EDITION 10.1d

File Edit View Compile Simulate Add Transcript Tools Layout

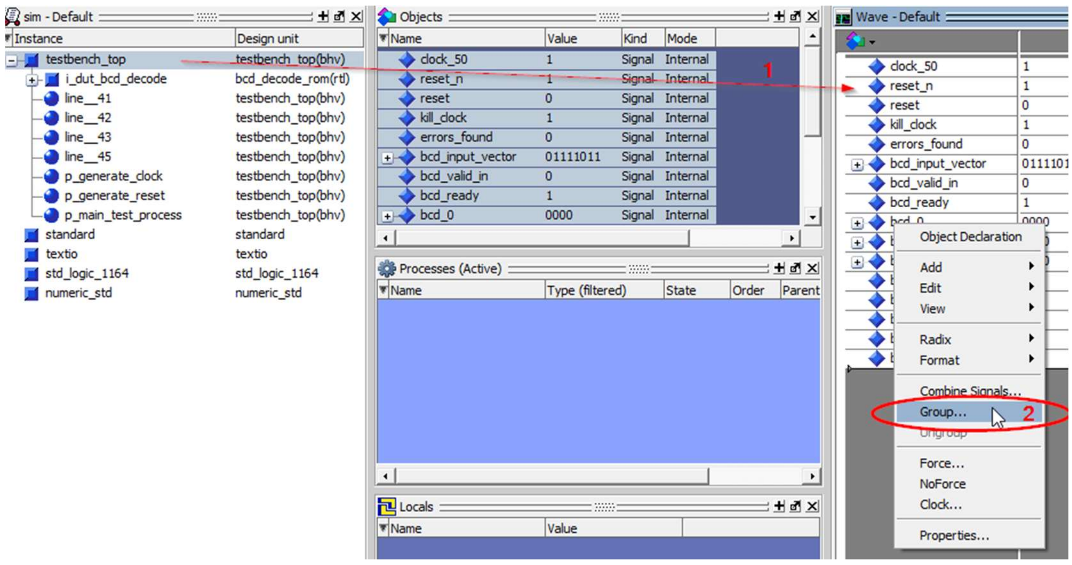
Layout Simulate

sim - Default

Instance	Design unit	Design unit type
testbench_top	testbench_top(bhv)	Architecture
i_dut_bcd_decode	bcd_decode_rom(rtl)	Architecture
line__41	testbench_top(bhv)	Process
line__42	testbench_top(bhv)	Process
line__43	testbench_top(bhv)	Process

Drag the signals of interest (preferably all of them), as you're used to into the wave window, group them to not mix them up.

You group signals by right clicking marked signals in the wave window and selecting “Group...”



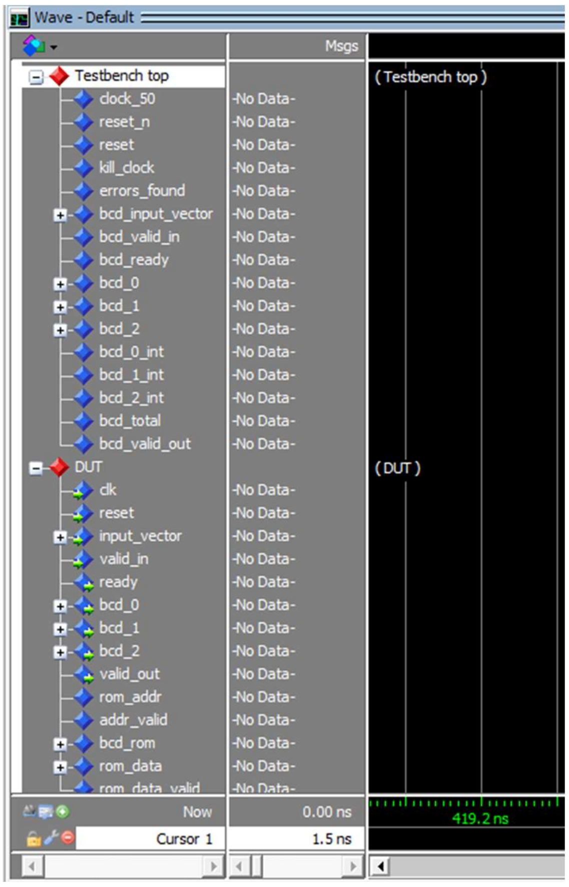
Instance	Design unit
testbench_top	testbench_top(bhv)
i_dut_bcd_decode	bcd_decode_rom(rtl)
line__41	testbench_top(bhv)
line__42	testbench_top(bhv)
line__43	testbench_top(bhv)
line__45	testbench_top(bhv)
p_generate_clock	testbench_top(bhv)
p_generate_reset	testbench_top(bhv)
p_main_test_process	testbench_top(bhv)
standard	standard
textio	textio
std_logic_1164	std_logic_1164
numeric_std	numeric_std

Name	Value	Kind	Mode
clock_50	1	Signal	Internal
reset_n	1	Signal	Internal
reset	0	Signal	Internal
kill_clock	1	Signal	Internal
errors_found	0	Signal	Internal
bcd_input_vector	01111011	Signal	Internal
bcd_valid_in	0	Signal	Internal
bcd_ready	1	Signal	Internal
bcd_0	0000	Signal	Internal

Processes (Active): Name, Type (filtered), State, Order, Parent

Locals: Name, Value

clock_50	1
reset_n	1
reset	0
kill_clock	1
errors_found	0
bcd_input_vector	0111101
bcd_valid_in	0
bcd_ready	1
bcd_0	0000



	Msgs
Testbench top	
clock_50	-No Data-
reset_n	-No Data-
reset	-No Data-
kill_clock	-No Data-
errors_found	-No Data-
bcd_input_vector	-No Data-
bcd_valid_in	-No Data-
bcd_ready	-No Data-
bcd_0	-No Data-
bcd_1	-No Data-
bcd_2	-No Data-
bcd_0_int	-No Data-
bcd_1_int	-No Data-
bcd_2_int	-No Data-
bcd_total	-No Data-
bcd_valid_out	-No Data-
DUT	
clk	-No Data-
reset	-No Data-
input_vector	-No Data-
valid_in	-No Data-
ready	-No Data-
bcd_0	-No Data-
bcd_1	-No Data-
bcd_2	-No Data-
valid_out	-No Data-
rom_addr	-No Data-
addr_valid	-No Data-
bcd_rom	-No Data-
rom_data	-No Data-
rom_data_valid	-No Data-
Now	0.00 ns
Cursor 1	1.5 ns

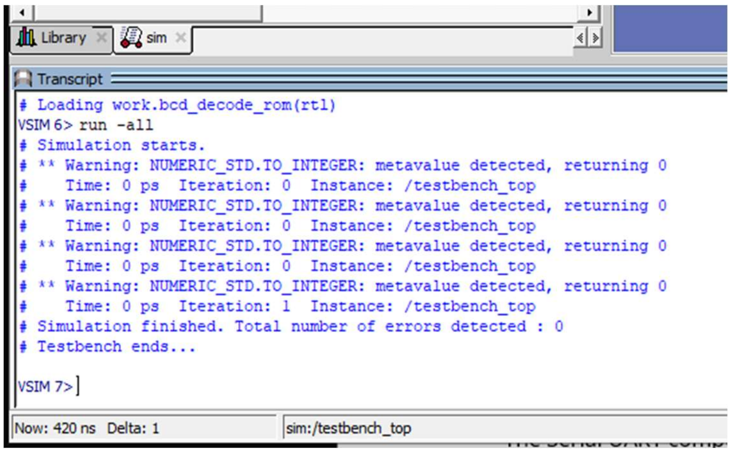
Name the group something smart, e.g. instance name, and repeat for all instances.

Run the full simulation with the command “run -all”

This command will run the testbench and make modelsim stop simulation when all clocks and sequential processes are stopped (see kill_clock signal in testbench file).

You will probably get a few warnings early in simulation, **this is normal and can safely be ignored.**

The transcript window should hopefully look something similar to the screenshot below:

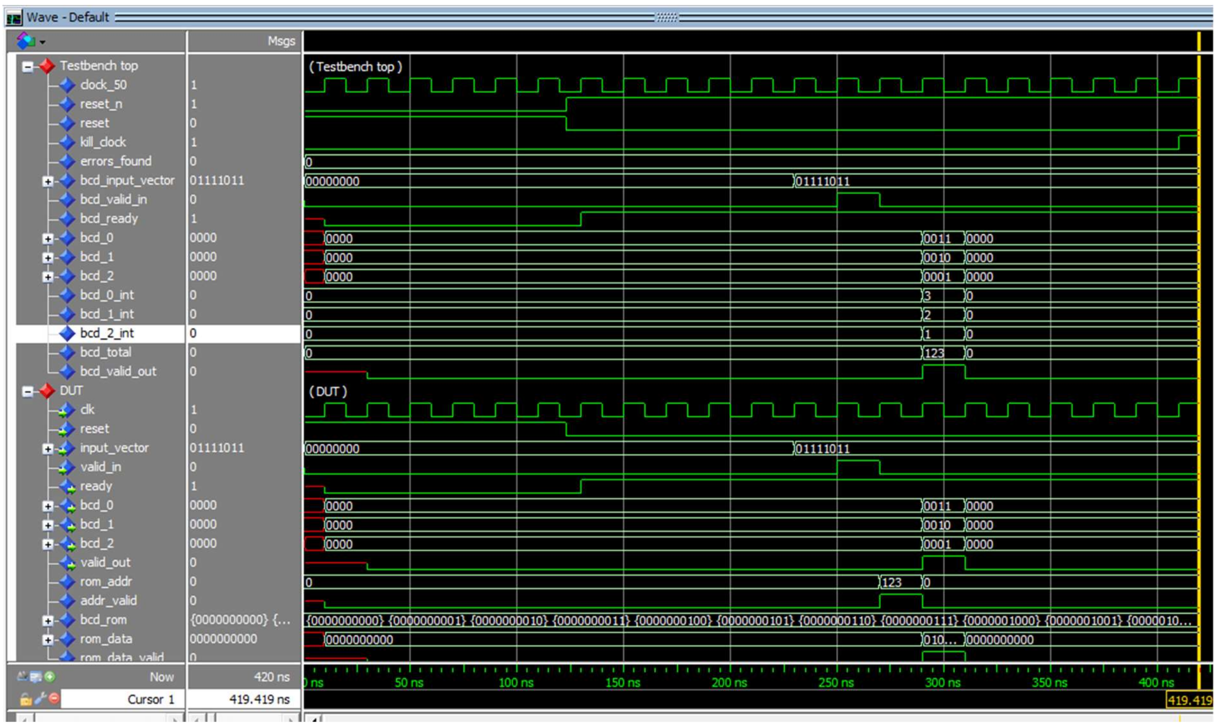


```
# Loading work.bcd_decode_rom(rtl)
VSIM 6> run -all
# Simulation starts.
# ** Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0
#    Time: 0 ps  Iteration: 0  Instance: /testbench_top
# ** Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0
#    Time: 0 ps  Iteration: 0  Instance: /testbench_top
# ** Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0
#    Time: 0 ps  Iteration: 0  Instance: /testbench_top
# ** Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0
#    Time: 0 ps  Iteration: 1  Instance: /testbench_top
# Simulation finished. Total number of errors detected : 0
# Testbench ends...

VSIM 7>
```

Now: 420 ns Delta: 1   sim:/testbench_top

Wave window should be updated and look similar to below:


Wave - Default
Msgs
Testbench top
clock_50 1
reset_n 1
reset 0
kill_clock 1
errors_found 0
bcd_input_vector 01111011
bcd_valid_in 0
bcd_ready 1
bcd_0 0000
bcd_1 0000
bcd_2 0000
bcd_0_int 0
bcd_1_int 0
bcd_2_int 0
bcd_total 0
bcd_valid_out 0
DUT
clk 1
reset 0
input_vector 01111011
valid_in 0
ready 1
bcd_0 0000
bcd_1 0000
bcd_2 0000
valid_out 0
rom_addr 0
addr_valid 0
bcd_rom {0000000000} {...
rom_data 0000000000
rom_data_valid 0
(Testbench top)
00000000
01111011
0011
0010
0001
3
2
1
123
(DUT)
{0000000000} {0000000001} {0000000010} {0000000011} {0000000100} {0000000101} {0000000110} {0000000111} {0000001000} {0000001001} {0000010...
010...
Now 420 ns
Cursor 1 419.419 ns
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350 ns
400 ns
419.419