Zürcher Hochschule
für Angewandte Wissenschaften

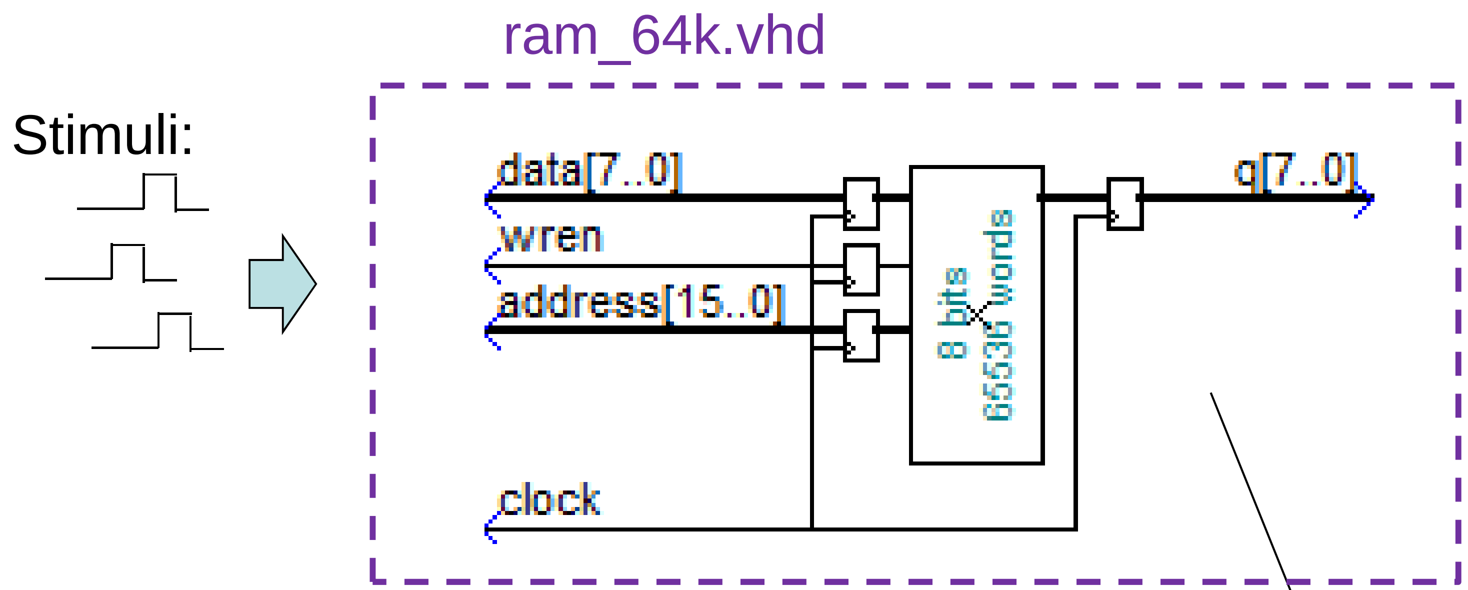
Test Benches

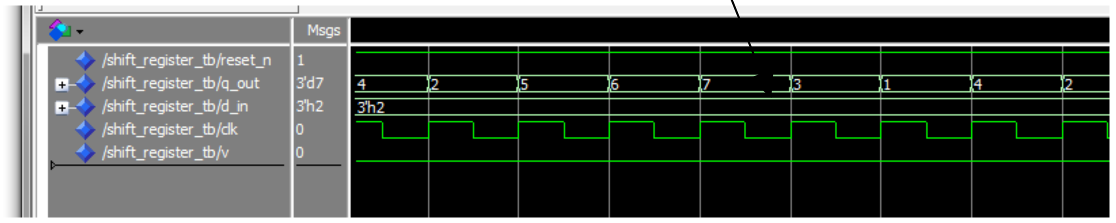
Zürcher Hochschule
für Angewandte Wissenschaften

Matroschka Doll

Test Bench for RAM

ram_tb.vhd

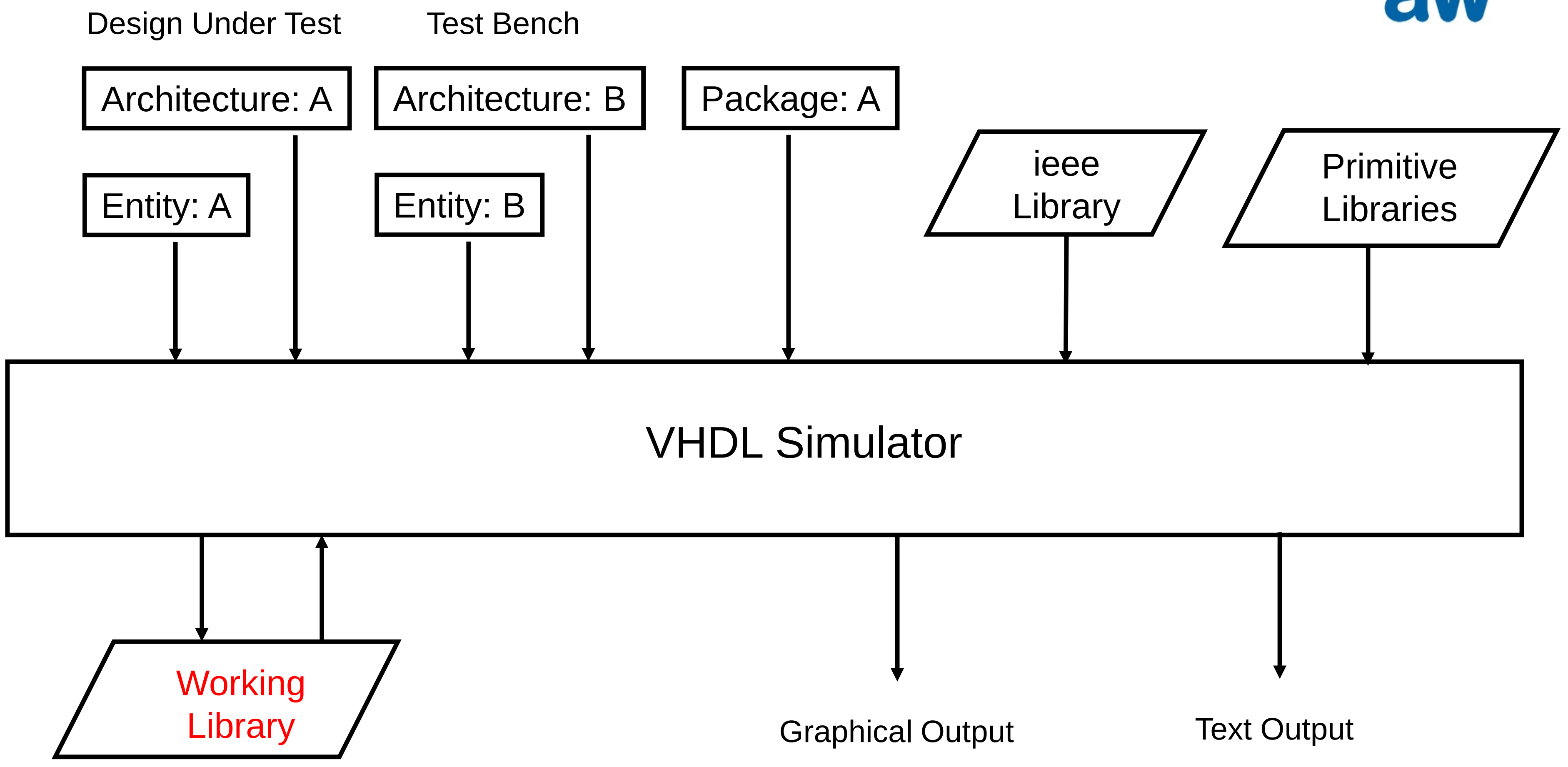




```
/shift_register_tb/reset_n   1
/shift_register_tb/q_out     3'd7   4  2  5  6  7  3  1  4  2
/shift_register_tb/d_in      3'h2   3'h2
/shift_register_tb/clk       0
/shift_register_tb/v         0
```

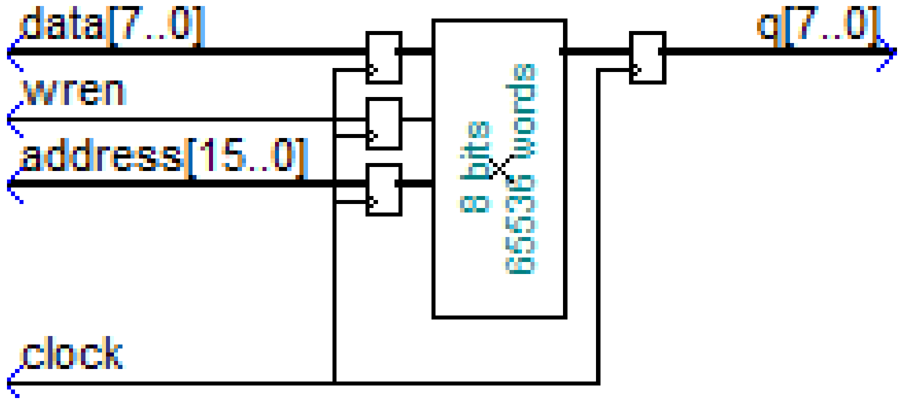
Graphical Simulation Results

VHDL Simulator



Entity of RAM_64k

ram64k.vhd

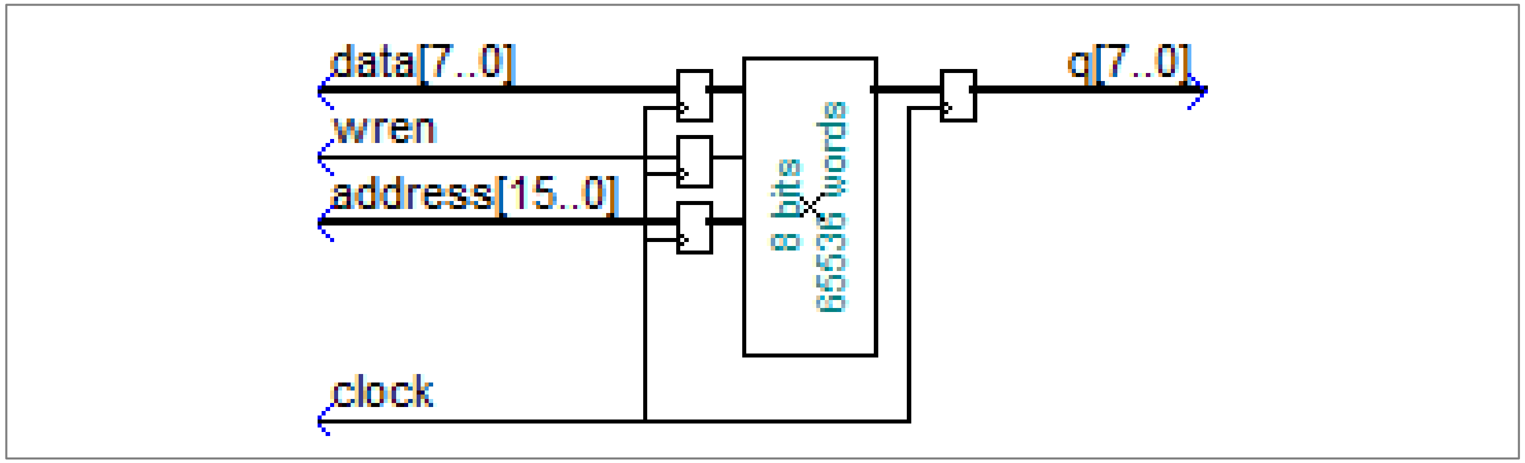


```
ENTITY ram_64k IS
  PORT(
        clock    : IN     std_logic;
        wren     : IN     std_logic;
        data     : IN     std_logic_vector(7 downto 0);
        q        : OUT    std_logic_vector(7 downto 0)
        );
END ram_64k;
```

Zürcher Hochschule
für Angewandte Wissenschaften

Test Bench: Component

ram_tb.vhd

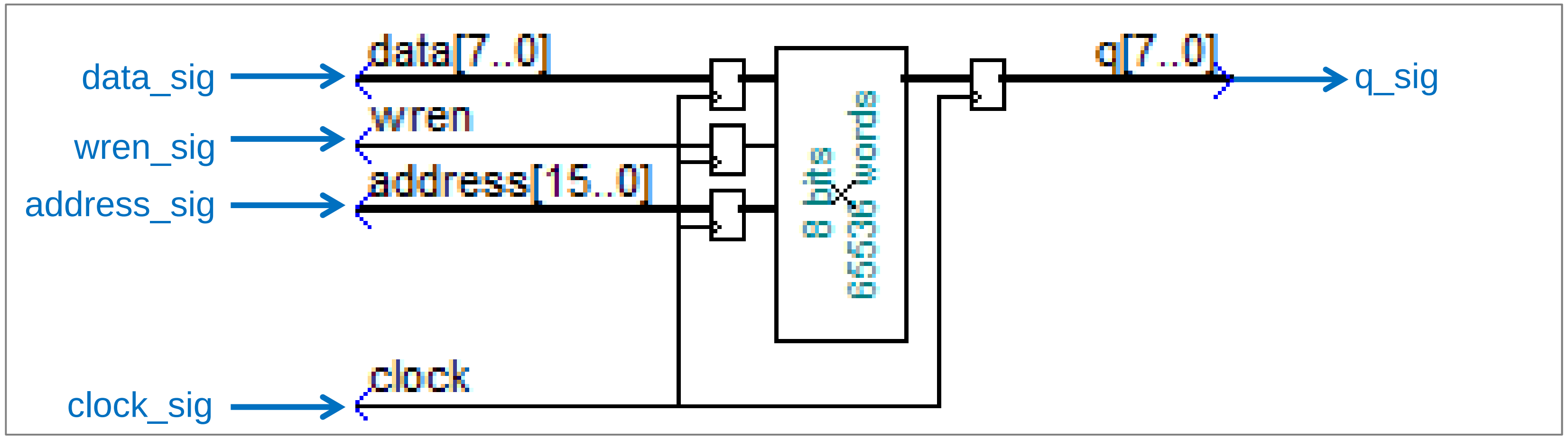


```
ARCHITECTURE struct OF ram_tb IS

COMPONENT ram_64k
          PORT(
          clock     : IN      std_logic;
          wren      : IN      std_logic;
          data      : IN      std_logic_vector(7 downto 0);
          q         : OUT     std_logic_vector(7 downto 0)
          );
END COMPONENT;


BEGIN
```

Zürcher Hochschule
für Angewandte Wissenschaften

Test Bench: Test Signals

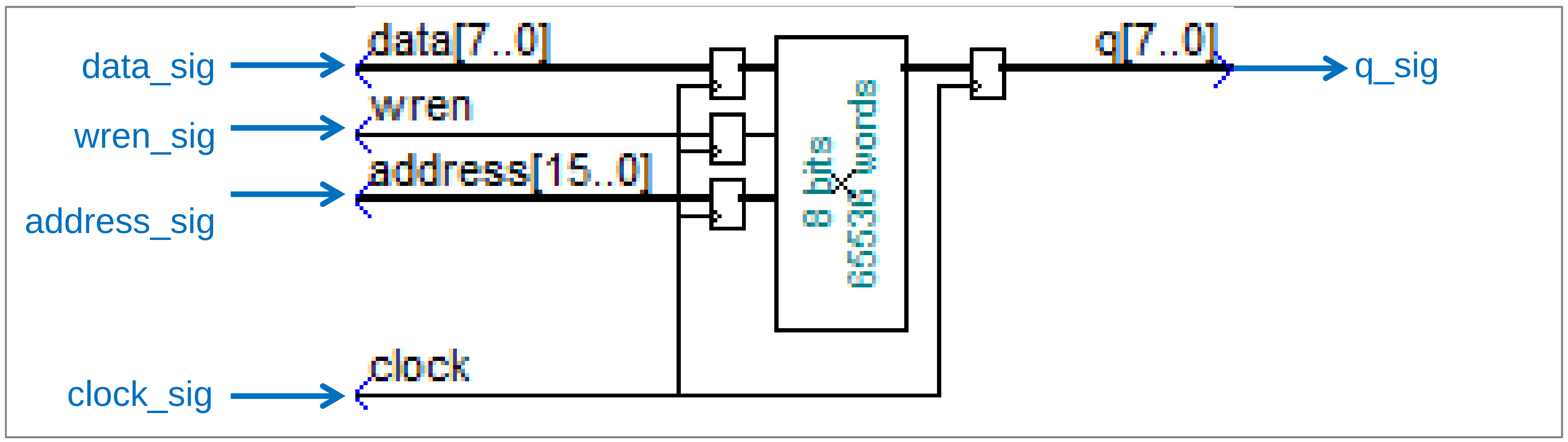
ram_tb.vhd



```
ARCHITECTURE struct OF ram_tb IS

COMPONENT top_level
          PORT( clock           : IN  std_logic;
                rst_n           : IN   std_logic;
                start           : IN   std_logic;
                segmente        : OUT  std_logic_vector(6 downto 0));
          END COMPONENT;

          SIGNAL    address_sig         : STD_LOGIC_VECTOR (15 DOWNTO 0);
          SIGNAL    clock_sig           : STD_LOGIC;
          SIGNAL    data_sig            : STD_LOGIC_VECTOR (7 DOWNTO 0);
          SIGNAL    wren_sig            : STD_LOGIC;
          SIGNAL    q_sig               : STD_LOGIC_VECTOR (7 DOWNTO 0);
BEGIN
```

Test Bench: Component Instantiation

ram_tb.vhd



```
BEGIN
  ram64k_inst : ram64k
  PORT MAP (
                address => address_sig,
                clock     => clock_sig,
                data      => data_sig,
                wren      => wren_sig,
                q         => q_sig
          );
```

Alternative Methode

```
ram64k_inst : ram64k
port map(address_sig, clock_sig, data_sig, wren_sig, q_sig);
```

Test Bench: Clock Generation

```
…
SIGNAL  clk_halfp       : time := 20ns;
…

    clkgen  : PROCESS
    BEGIN
        clock_signal <= '0';
        WAIT FOR 1*clk_halfp;
        clock_signal <= '1';
        WAIT FOR 1*clk_halfp;
    END PROCESS clkgen;
…
END struct;
```

Assert Statement

```
ASSERT condition REPORT string SEVERITY SEVERITY_level ;
```

Falls „Condition“ nicht erfüllt,
wird ein Report generiert

Mögliche level sind:
note
warning
ERROR
failure (bricht Simulation ab)

Beispiele:

```
ASSERT (A = B) REPORT “A ungleich B“ SEVERITY ERROR ;

ASSERT false REPORT “Test programm beendet“ SEVERITY note ;
```

- „ASSERT“ erlaubt bei einer bestimmten Bedingung im Simulations-programm einen Bericht auszugeben oder das Simulationsprogramm ganz zu stoppen.
- Assert wird bei der Synthese ignoriert.

Zürcher Hochschule
für Angewandte Wissenschaften

Example: Program to test bcd-gray converter

Gibt nur eine Fehlermeldung aus,
Beendet aber die Simulation nicht
(falls sie hier failure einsetzen wird
die Simulation vorzeitig gestoppt und sie
sehen die folgenden
Simulationsergebnisse nicht mehr)

```
STIMULUS: process
begin
   bcd <= "000" ;
   WAIT FOR 10 ns;
   ASSERT (gray = "000") REPORT "expected "000" " SEVERITY ERROR;
   WAIT FOR 100 ns;

   bcd <= "111";
   WAIT FOR 10 ns;
   ASSERT (gray = "100") REPORT "expected „100“ " SEVERITY ERROR;
   WAIT FOR 100 ns;

   ASSERT false REPORT " --- ALL TESTS PASS ---" SEVERITY failure;

end process;
```

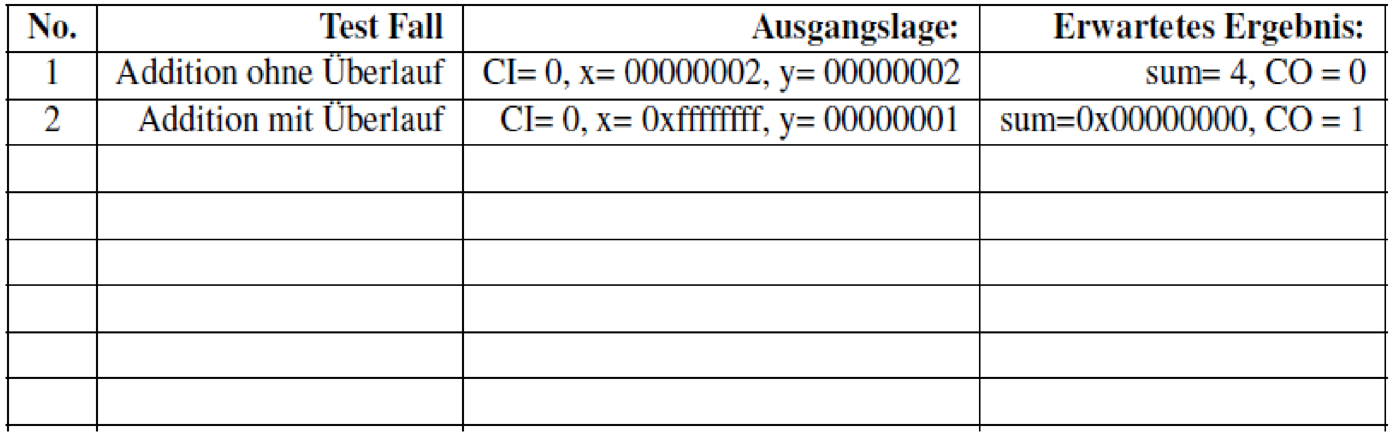
Beendet die Simulation

Zürcher Hochschule
für Angewandte Wissenschaften

Text File Based Simulation

Test Plan

- At the time of specification, also write a test plan
- Before you write a test bench, you must have a test plan
- Best is, if someone else, but the designer writes test plan

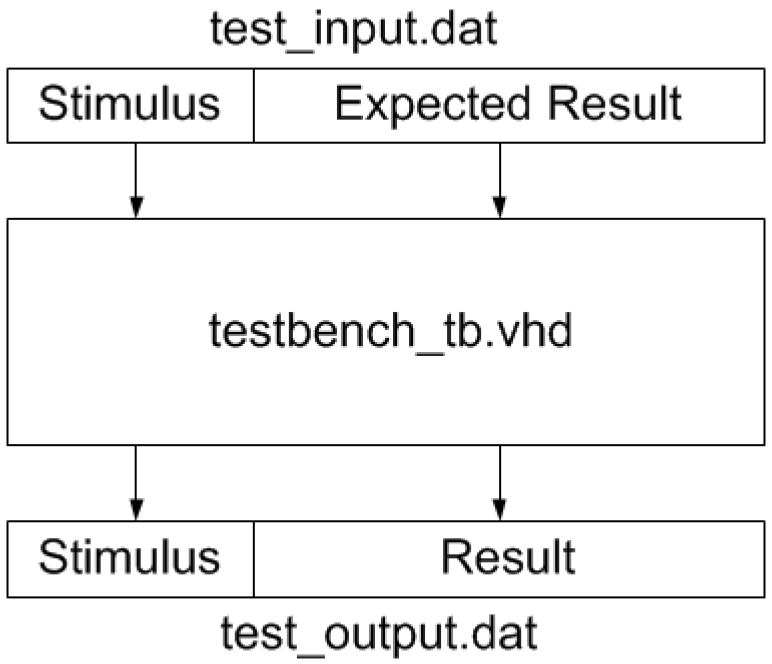


No.	Test Fall	Ausgangslage:	Erwartetes Ergebnis:
1	Addition ohne Überlauf	CI= 0, x= 00000002, y= 00000002	sum= 4, CO = 0
2	Addition mit Überlauf	CI= 0, x= 0xffffffff, y= 00000001	sum=0x00000000, CO = 1

Text Based Test Command File

```
CI Summand1   Summand2     Sum    CO
0 00000001 00000001 00000002 0
0 00000002 00000002 00000004 0
0 00000004 00000004 00000008 0
0 FFFFFFFF FFFFFFFF FFFFFFFE 1
0 0000AAAA AAAA0000 AAAAAAAA 0
0 158D7129 E4C28B56 FA4FFC7F 0
1 00000001 00000001 00000003 0
```

Textfile based Test Vectors



Entity of Testbench

- std.textio.all is needed to read text files

```
library ieee;
use ieee.std_logic_1164.all;
use work.all;
use std.textio.all;
use work.std_logic_textio.all;
```

- Testbench Entity has no I/O

```
entity adder32_tb is
end adder32_tb;
```

- Testbench Entity is Top Level

Variables used in the following examples

File «variable» of Type «Text»

```
readcmd: PROCESS
FILE cmdfile: TEXT;        -- Define the file 'handle'
FILE outfile: TEXT;        -- Define the file 'handle'
VARIABLE line_in,line_out: Line; -- Line buffers
VARIABLE good: boolean;    -- Status of the read operations
VARIABLE reset_n_ti: std_logic;
VARIABLE hcount_ti: integer;
VARIABLE hsync_ti: std_logic;
VARIABLE vcount_ti: integer;
VARIABLE vsync_ti: std_logic;
CONSTANT hline_length: integer := 1344;
BEGIN
```

Variables of Type «Line»
for temporal storage of lines

Command Files Include Vectors and Results

```
CI Summand1   Summand2      Sum   CO
0 00000001 00000001 00000002 0
0 00000002 00000002 00000004 0
0 00000004 00000004 00000008 0
0 FFFFFFFF FFFFFFFF FFFFFFFE 1
0 0000AAAA AAAA0000 AAAAAAAA 0
0 158D7129 E4C28B56 FA4FFC7F 0
1 00000001 00000001 00000003 0
```

VHDL command: FILE_OPEN
VHDL command: readline

Name and path to text file

Open file for read or write mode

Read a line from file and fill buffer «line_in»

```
FILE_OPEN(cmdfile,"testcase_1_video_contr.dat",READ_MODE);
FILE_OPEN(outfile,"testcase_1_results.dat",WRITE_MODE);
      LOOP
            IF endfile(cmdfile) then  -- Check EOF
                assert false
                    report "End of test case encountered;exiting."
                    severity NOTE;
                EXIT;
            END IF;
        readline(cmdfile,line_in);      -- Read a line from the file
        NEXT WHEN line_in'length = 0;   -- Skip empty lines
```

VHDL command read oder hread (vector)

Variable reset_n_ti
is loaded with
arg1 of line

Variable «good» is set
when read successfull

Linebuffer

```
read(line_in,reset_n_ti,good);-- Read the reset_n inp
      assert good
      report "Text I/O read error”
      severity ERROR;
read(line_in,hcount_ti,good);
      assert good
        report "Text I/O read error"
        severity ERROR;
```

Variable hcount_ti
is loaded with
arg 2 of line

VHDL Command write oder hwrite (vector)

```
 wait until falling_edge(clk);

                  cin <= CI;
                  x <= A;
                  y <= B;

         wait for 2 * PERIOD;    -- Give the circuit time to stabilize
         if (sum = S) then
                  write(line_out,string'("Test passed:"));
         else
                  write(line_out,string'("Test FAILED:"));
         end if;

         hwrite(line_out,sum,RIGHT,9);
         writeline(outfile,line_out);      -- write the message
     end loop;
     wait;

end process;
```

Fill Buffer «line_out» with string «Test passed»

Fill Buffer «line_out» with Variable «sum»

Write data from Buffer «line_out»

Output Data in File

```
testexecution: PROCESS

FILE outfile: TEXT;       -- Define the file 'handle‘

VARIABLE line_out: Line; -- Line buffers

BEGIN

FILE_OPEN(outfile,"testcase_1_results.dat",WRITE_MODE);

        LOOP

        write(line_out, string'(integer'image(lincnt)));

        write(line_out, string'("Reset Sync polarity wrong"));

        writeline(outfile,line_out);

        END LOOP;

WAIT;

END PROCESS;
```

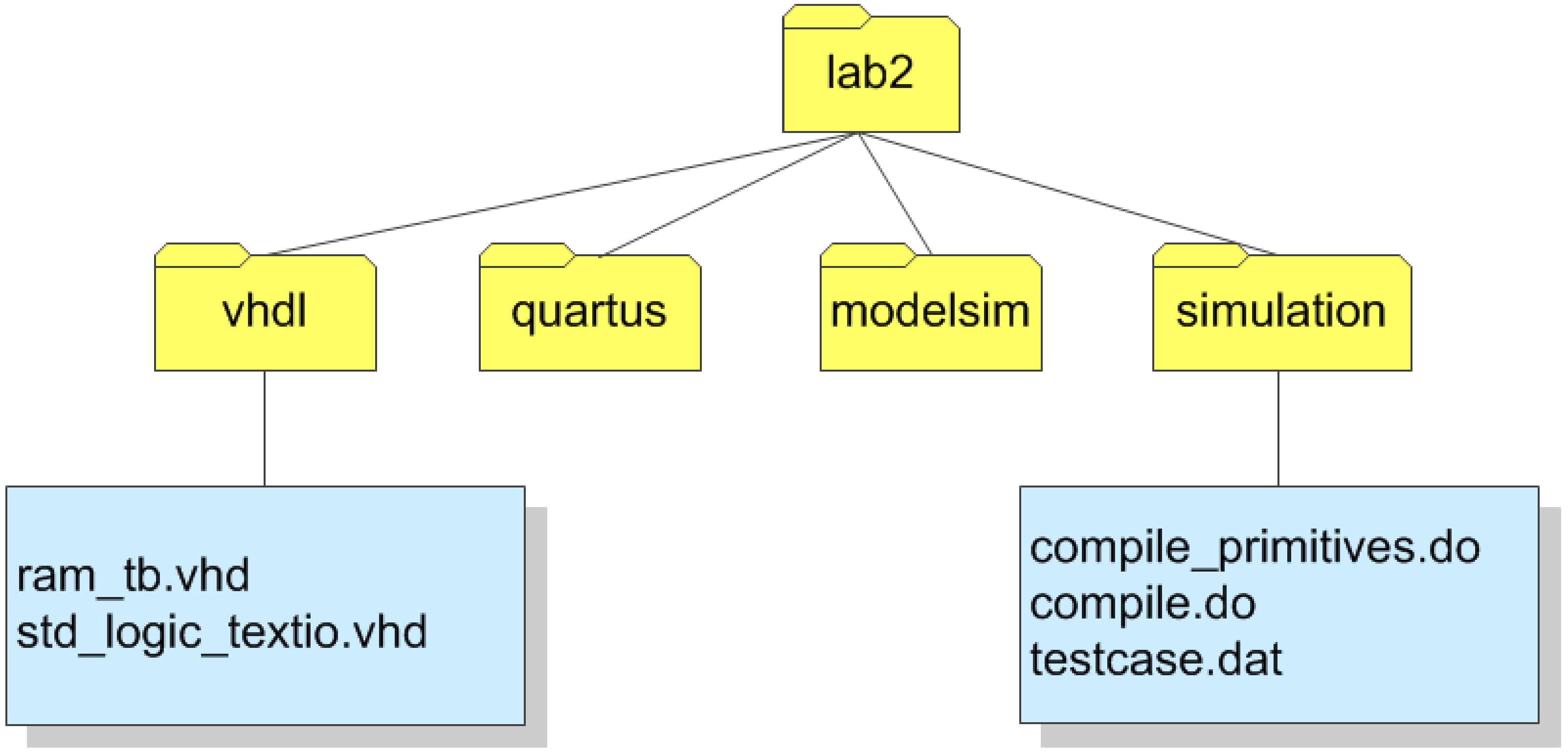
Writes Output to Transcript Window

```
Alternativ: writeline(OUTPUT,line_out);
```

Output Result

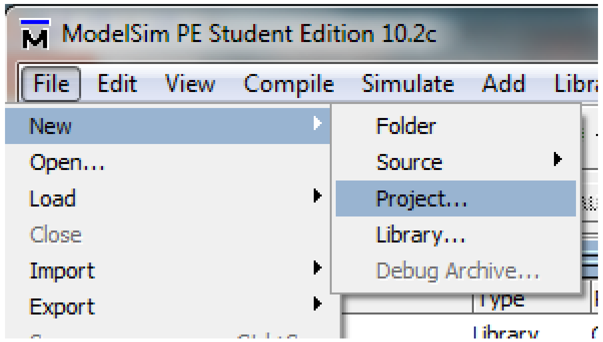
```
# Test passed: 0 00000001 00000001 00000002 0
# Test passed: 0 00000002 00000002 00000004 0
# Test passed: 0 00000004 00000004 00000008 0
# Test passed: 0 FFFFFFFF FFFFFFFF FFFFFFFE 1
# Test passed: 0 0000AAAA AAAA0000 AAAAAAAA 0
# Test passed: 0 158D7129 E4C28B56 FA4FFC7F 0
# Test passed: 1 00000001 00000001 00000003 0
# Test passed: 1 A4F67B92 00000001 A4F67B94 0
# Test passed: 1 FFFFFFFF FFFFFFFE FFFFFFFE 1
# Test passed: 1 FFFFFFFE FFFFFFFF FFFFFFFE 1
# Test passed: 1 00000002 00000004 00000007 0
```

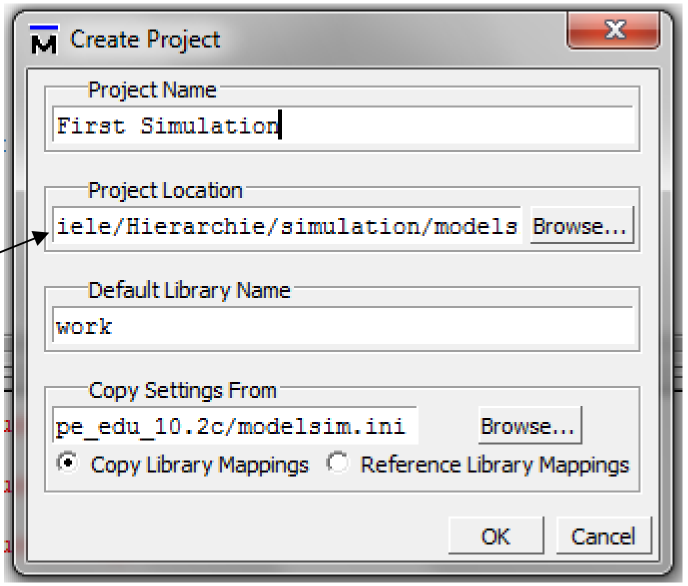
File Structure for Simulation



Start Simulator

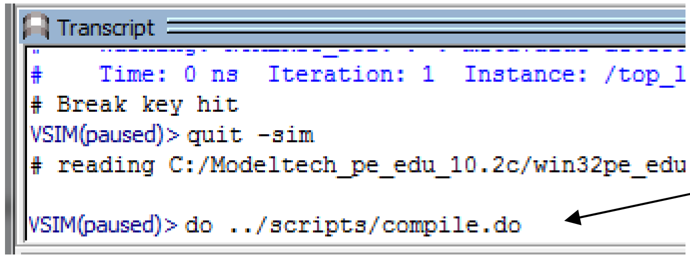
Creating a new Project





Project Name: First Simulation
Project Location: iele/Hierarchie/simulation/models
Default Library Name: work
Copy Settings From: pe_edu_10.2c/modelsim.ini
[✓ Copy Library Mappings] Reference Library Mappings

Arbeitsverzeichnis
ins Projektverzeichnis
legen



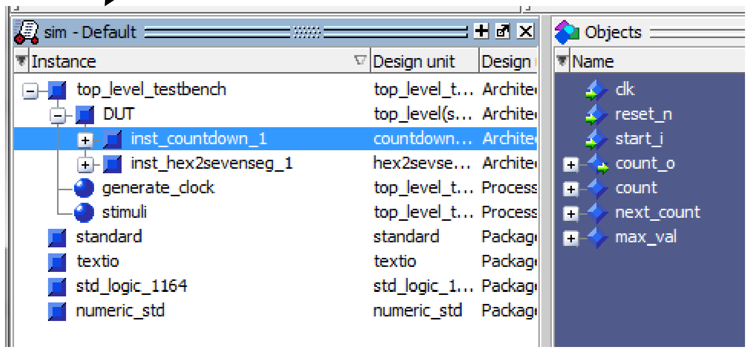
```
#    Time: 0 ns  Iteration: 1  Instance: /top_l
# Break key hit
VSIM(paused)> quit -sim
# reading C:/Modeltech_pe_edu_10.2c/win32pe_edu

VSIM(paused)> do ../scripts/compile.do
```

Simulation von Kommandozeile starten

Waveform Window

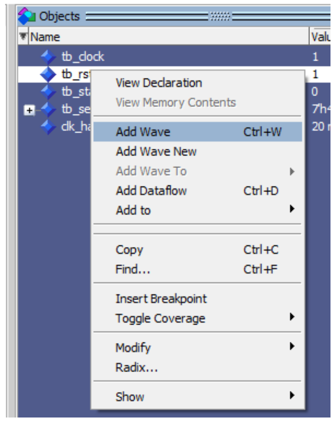
Structure Window
(Design Hierarchie Browser)

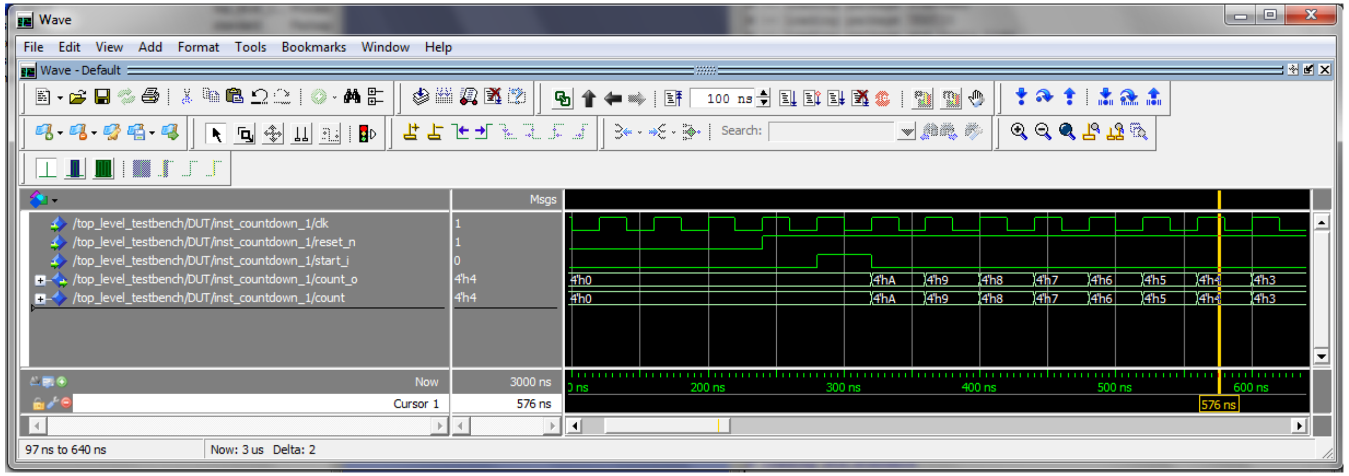
Objects Window
(zu simulierende Signale)



Instance	Design unit	Design
top_level_testbench	top_level_t…	Archite
DUT	top_level(s…	Archite
inst_countdown_1	countdown…	Archite
inst_hex2sevenseg_1	hex2sevse…	Archite
generate_clock	top_level_t…	Process
stimuli	top_level_t…	Process
standard	standard	Packag
textio	textio	Packag
std_logic_1164	std_logic_1…	Packag
numeric_std	numeric_std	Packag

Add Wave



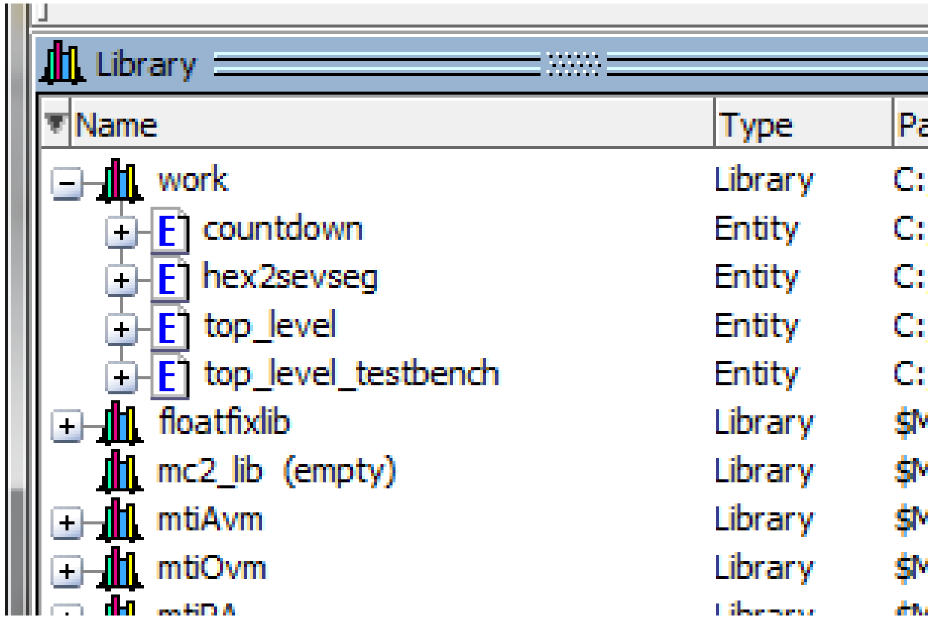


	Msgs
/top_level_testbench/DUT/inst_countdown_1/clk	1
/top_level_testbench/DUT/inst_countdown_1/reset_n	1
/top_level_testbench/DUT/inst_countdown_1/start_i	0
/top_level_testbench/DUT/inst_countdown_1/count_o	4'h4
/top_level_testbench/DUT/inst_countdown_1/count	4'h4
Now	3000 ns
Cursor 1	576 ns

97 ns to 640 ns   Now: 3 us Delta: 2

Zürcher Hochschule
für Angewandte Wissenschaften

Library Window



Library

Name	Type	Pa
work	Library	C:
countdown	Entity	C:
hex2sevseg	Entity	C:
top_level	Entity	C:
top_level_testbench	Entity	C:
floatfixlib	Library	$M
mc2_lib (empty)	Library	$M
mtiAvm	Library	$M
mtiOvm	Library	$M