DTP2_VHDL_0:

Warm-up VHDL Inhalt von DTP1

- VHDL Block Description
  - Entity / Architecture / RTL Diagram
  - Ports / Signals / Process for FF or Reg- and Comb-Logic

Zürcher Hochschule
für Angewandte Wissenschaften

DTP2_VHDL_1:

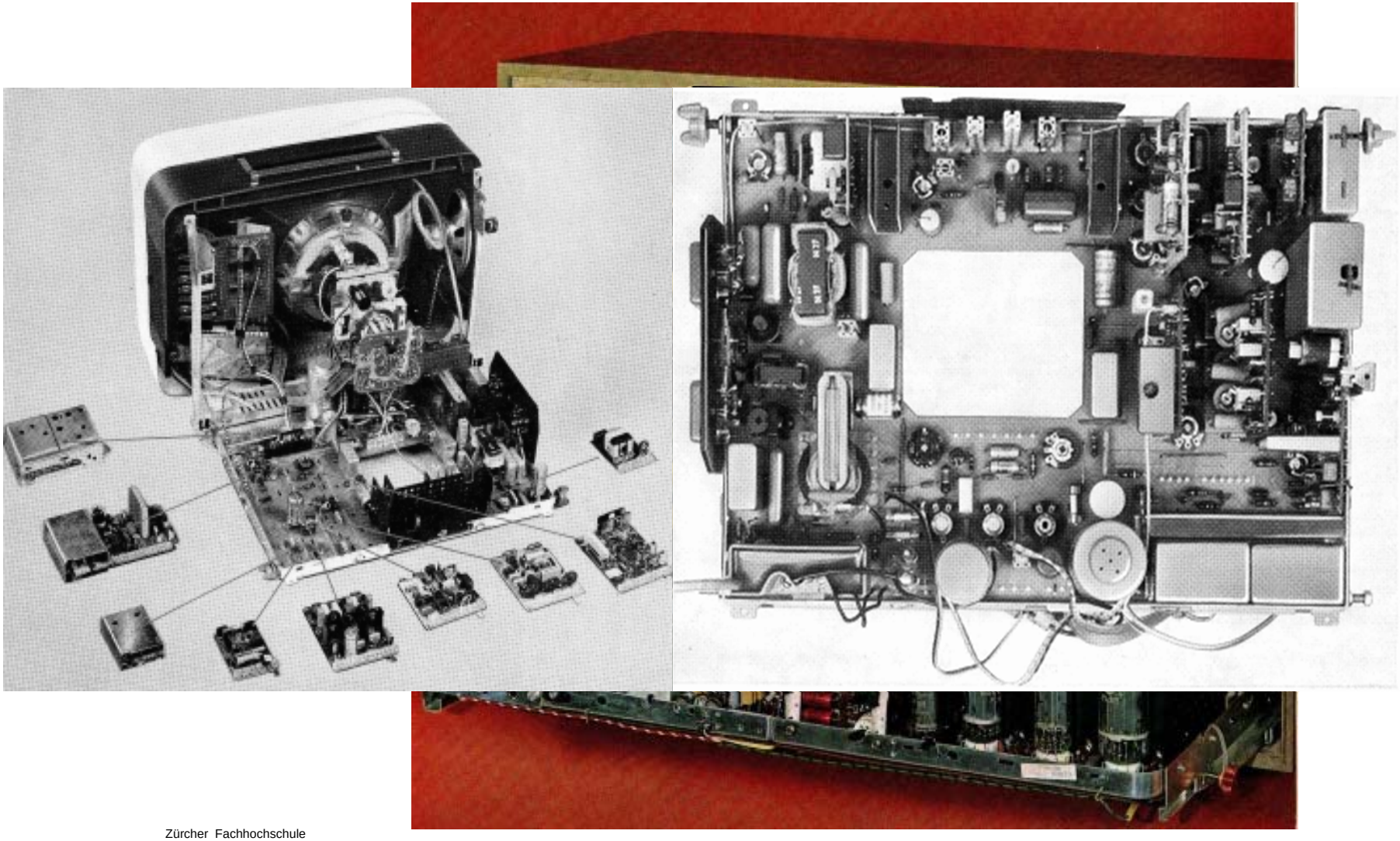
Hierarchisches VHDL und Simulation

- Aufbau Hierarchisches Design
  - Component Declaration and Instantiations
- Testbench
  - Device-Under-Test, Stimuli, Clock-Generator, Checks
- Simulation
  - Tools, Libraries und Script
- Mini-Übung
  - Analyse des Code Beispiels einfach_schaltung.vhd (LAB1_source_files) und zeichnen des RTL Diagramms

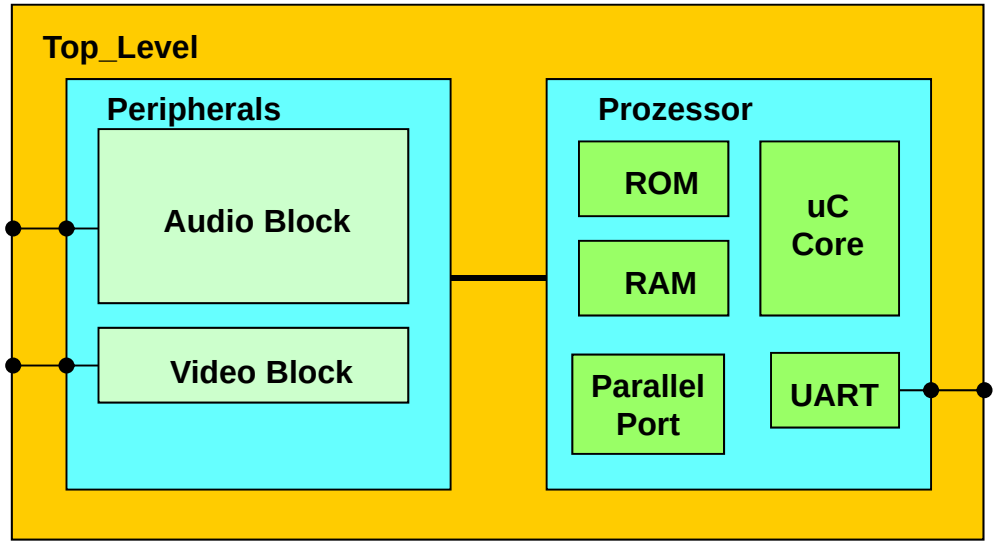
Zürcher Hochschule
für Angewandte Wissenschaften

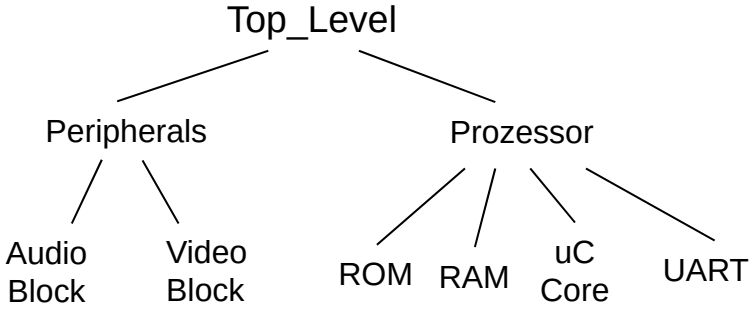
Hierarchisches VHDL Design

Hierarchisches Design



Hierarchisches VHDL Design oder Strukturale Modelierung

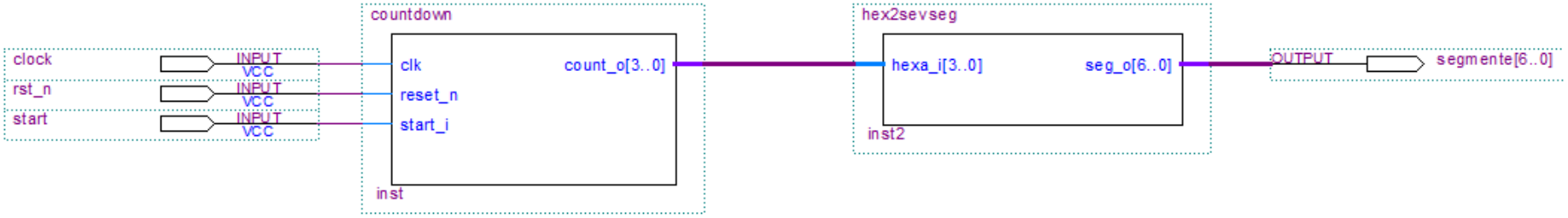




Matroschka Puppe

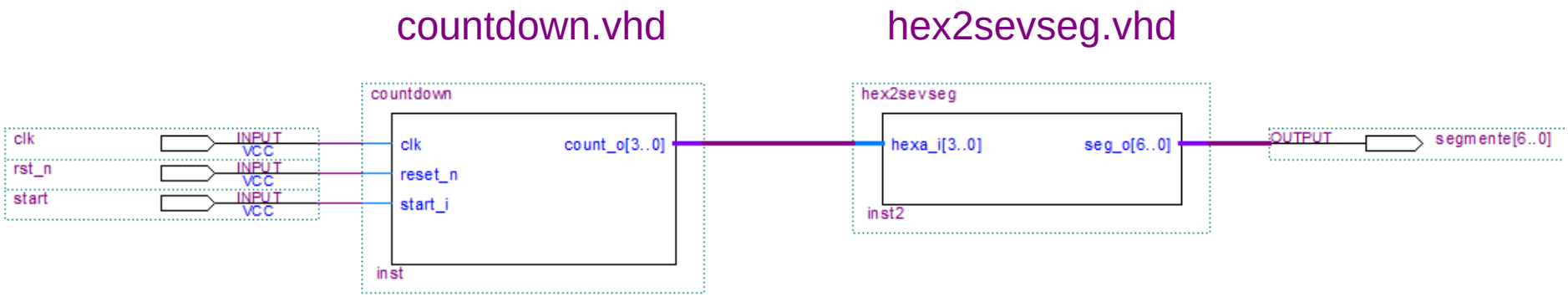
Beispiel für ein Hierarchisches Design

countdown.vhd hex2sevseg.vhd



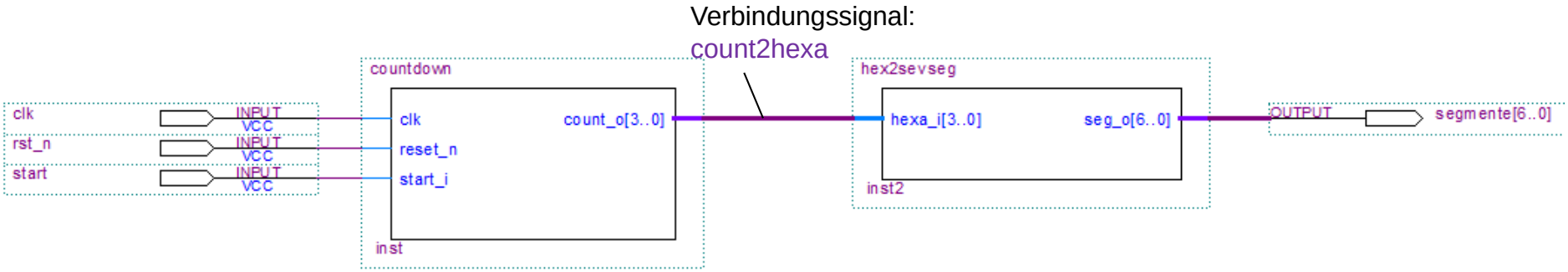
Dieser Top Level Schaltplan soll durch VHDL Code ersetzt werden

1. Der Top Level erhält eine Entity




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

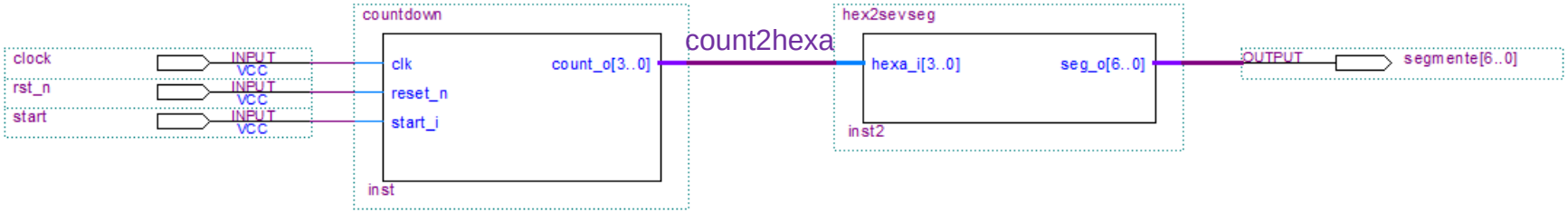
2. Der Top Level erhält eine Architektur mit Component Declarations




```
ARCHITECTURE struct OF top_level IS
        -- components and signals declaration
        COMPONENT hex2sevseg
        PORT( hexa_i      : IN   std_logic_vector(3 downto 0);
              seg_o       : OUT  std_logic_vector(6 downto 0)
            );
        END COMPONENT;

        COMPONENT countdown
        PORT( clk,
              reset_n              : IN    std_logic;
              start_i              : IN    std_logic;
              count_o              : OUT   std_logic_vector(3 downto 0)
             );
        END COMPONENT;

        SIGNAL count2hexa          : std_logic_vector(3 downto 0);
BEGIN
```

3. Die Komponenten werden auf dem Top Level instanziiert

Instance = Realisierung einer Datengruppe



Entity Name

Signal Namen in der übergeordneter Architektur

Instance Name (frei wählbar)

```
BEGIN
  inst_countdown_1: countdown
  PORT MAP (
            clk          =>      clock,
            reset_n      =>      rst_n,
            start_i      =>      start,
            count_o      =>      count2hexa
                 );

inst_hex2sevenseg_1: hex2sevseg
  PORT MAP (
            hexa_i       =>      count2hexa,
            seg_o        =>      segmente
            );

END struct;
```

Ports der untergeordneten Entity (siehe Component Declaration)

Komplettes Beispiel einer Top Level Architektur

Zürcher Hochschule für Angewandte Wissenschaften

```
ARCHITECTURE struct OF top_level IS

      COMPONENT hex2sevseg
      PORT(
      hexa_i      : IN   std_logic_vector(3 downto 0);
      seg_o       : OUT  std_logic_vector(6 downto 0));
      END COMPONENT;

      COMPONENT countdown
      PORT( clk,reset_n : IN    std_logic;
            start_i     : IN    std_logic;
            count_o     : OUT   std_logic_vector(3 downto 0));
      END COMPONENT;

      SIGNAL count2hexa : std_logic_vector(3 downto 0);
BEGIN

  inst_countdown_1: countdown
  PORT MAP (clk        =>    clock,
            reset_n    =>    rst_n,
            start_i    =>    start,
            count_o =>  count2hexa
            );

inst_hex2sevenseg_1: hex2sevseg
  PORT MAP (hexa_i     =>    count2hexa,
            seg_o =>    segmente
            );

END struct;
```

Übung1: Schaltplan der Hierarchie zeichnen

Zürcher Hochschule
für Angewandte Wissenschaften

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY dekoder_einfach IS PORT (
    A: IN std_logic;
    B: IN std_logic;
    C: IN std_logic;
    D: IN std_logic;
    Z: OUT std_logic );
END dekoder_einfach ;

ARCHITECTURE structural OF dekoder_einfach IS
    SIGNAL int1: std_logic;
    COMPONENT nand2
        PORT (  in1,in2 : IN std_logic;
                outp : OUT std_logic );
    END COMPONENT;
    COMPONENT and3
        PORT (  in1,in2,in3 : IN std_logic;
                outp         : OUT std_logic );
    END COMPONENT;
BEGIN
    inst1: nand2 PORT MAP (
        in1      => A,
        in2      => B,
        outp     => int1    );
    inst2: and3 PORT MAP (
        in1      => int1,
        in2      => C,
        in3      => D,
        outp     => Z );
END ARCHITECTURE structural;
```

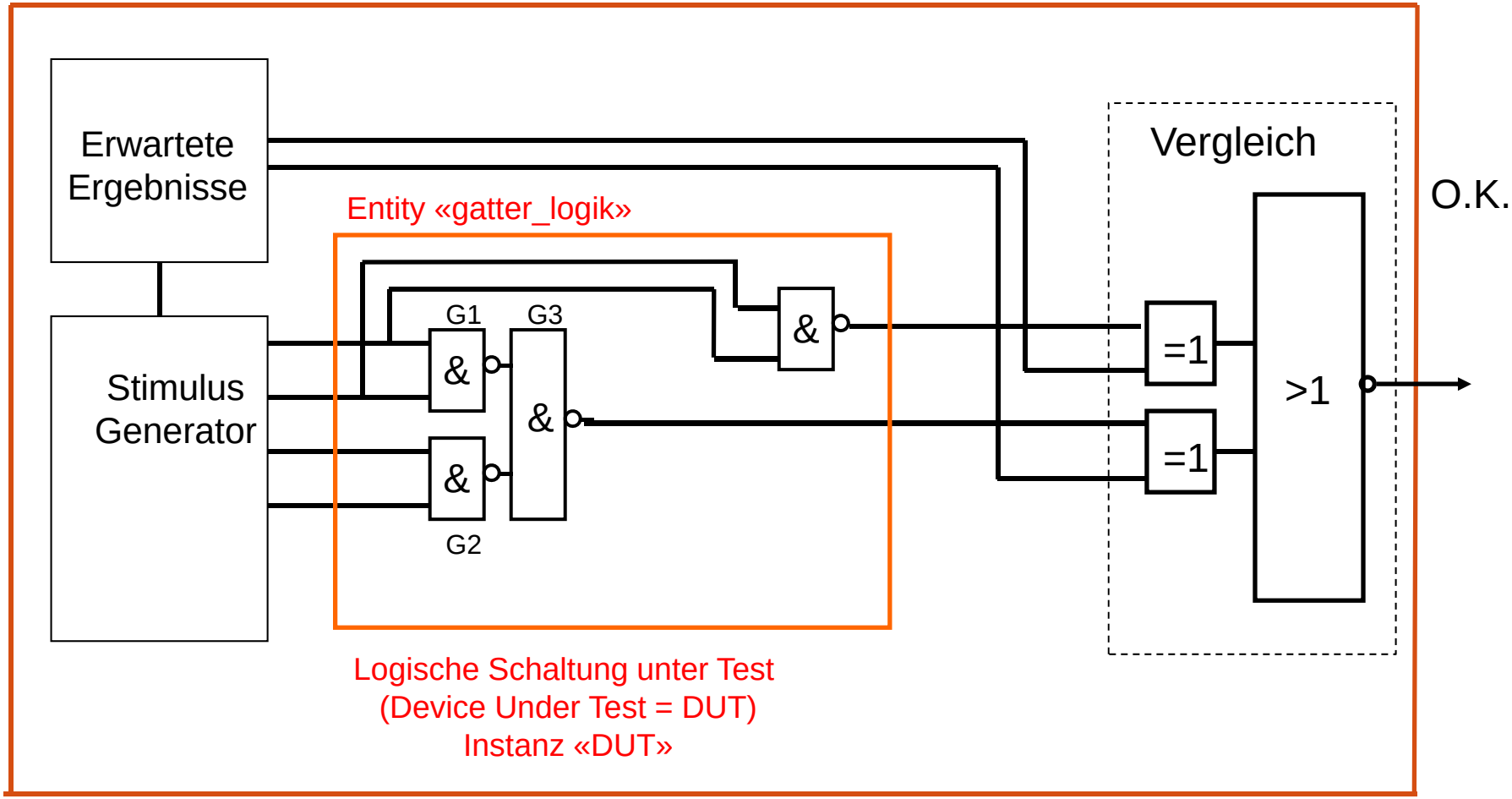
Zürcher Hochschule
für Angewandte Wissenschaften

VHDL Simulation

Simulatoren und Debugger
Tool = Modelsim

Zürcher Hochschule für Angewandte Wissenschaften

Entity «Testbench»



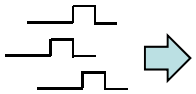
Testbench eine Stufe über Top_Level

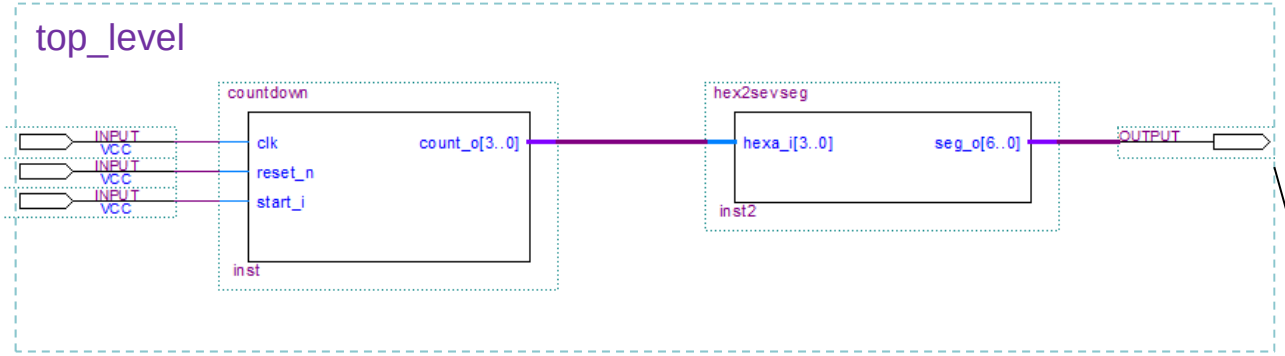
Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

top_level_testbench

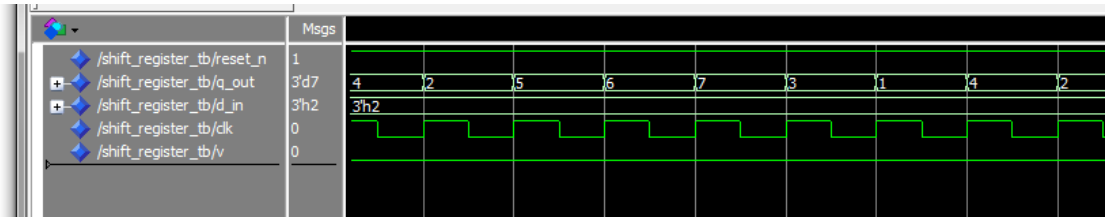
Stimuli:





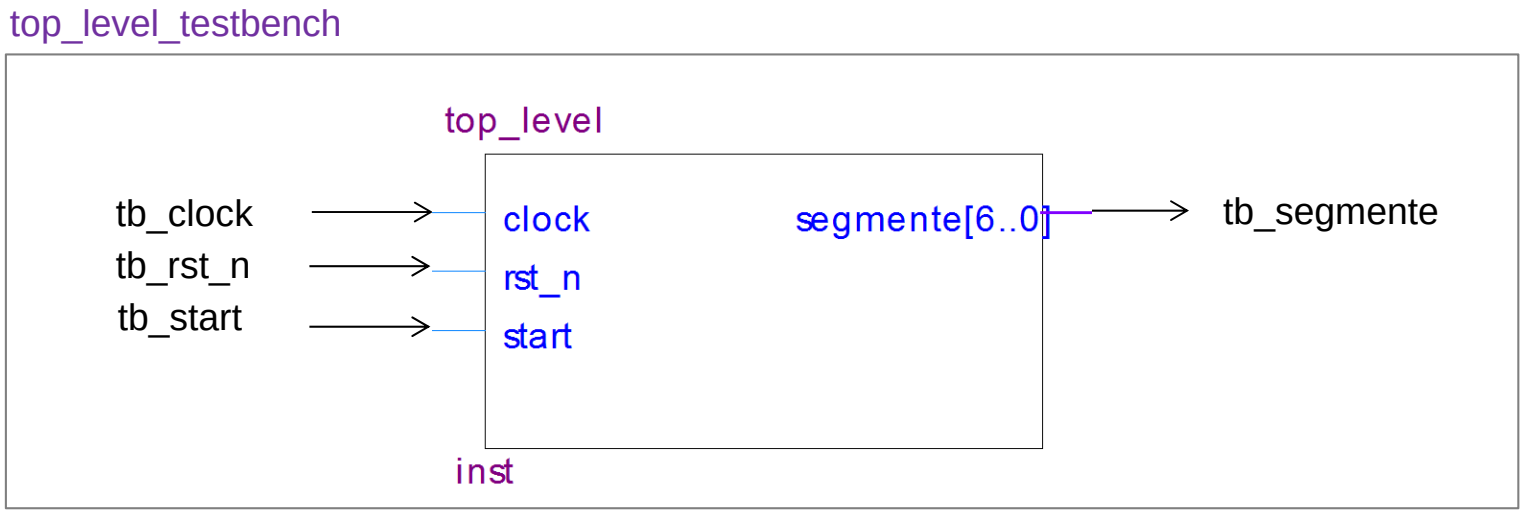
Visual
Check

Simulationsausgabe:



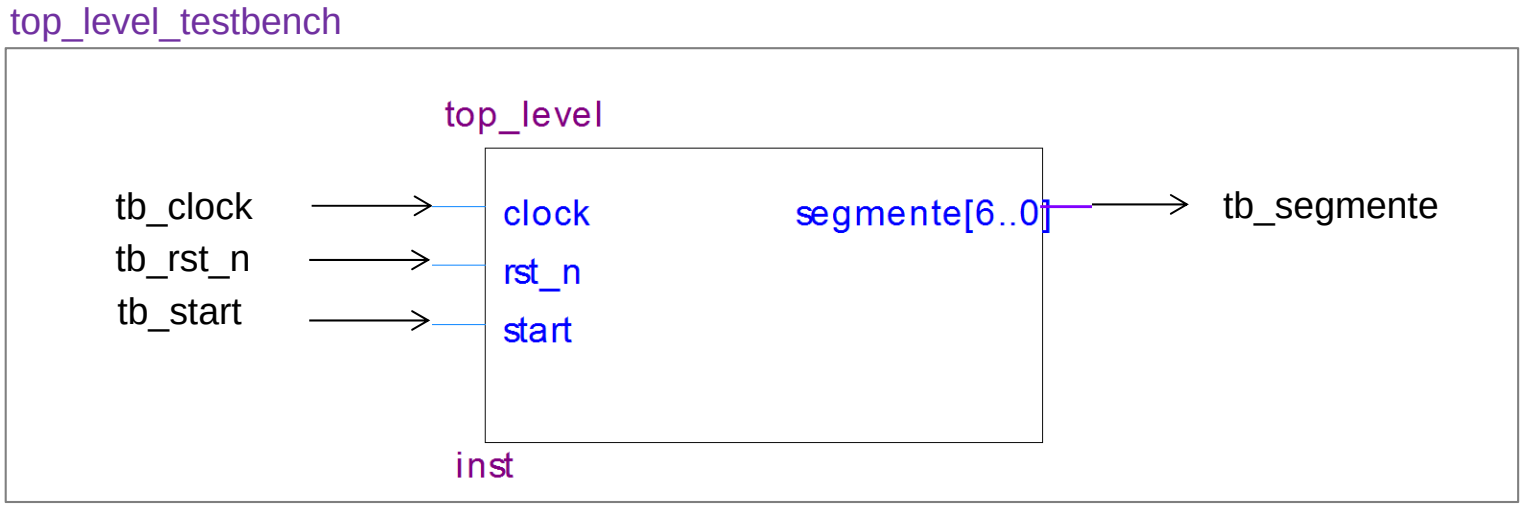

```
ENTITY testbench_counter IS

END testbench_counter;
```

Definition der zu testenden Komponente und Signale




```
ARCHITECTURE struct OF testbench_counter 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    tb_clock  : std_logic;
          SIGNAL    tb_rst_n  : std_logic;
          SIGNAL    tb_start  : std_logic;
          SIGNAL    tb_segmente         : std_logic_vector(6 downto 0);
          CONSTANT clk_halfp  : time := 20 ns;

BEGIN
```

Instanziierung von *top_level* als *Device Under Test* (DUT)




```
BEGIN

  DUT: top_level
  PORT MAP (     clock =>          tb_clock,
                 rst_n =>          tb_rst_n,
                 start =>          tb_start,
                 segmente =>       tb_segmente
           );
```

Erzeugung des Taktes für die Simulation

Zürcher Hochschule
für Angewandte Wissenschaften

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

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

(Ohne Sensitivity Liste)

Einfache Generierung von Stimuli

```
stimuli: PROCESS
         BEGIN
         tb_rst_n <= '0';
         tb_start <= '0';

         WAIT FOR 12 * clk_halfp;
         tb_rst_n <= '1';

         WAIT FOR 2 * clk_halfp;
         tb_start <= '1';

         WAIT FOR 2 * clk_halfp;
         tb_start <= '0';

         WAIT FOR 20 * clk_halfp;
         tb_start <= '1';

         WAIT FOR 2 * clk_halfp;
         tb_start <= '0';

         WAIT;
         END PROCESS stimuli;

END struct;
```

Wait Statements

Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

`WAIT UNTIL condition;`	Warten bis ein bestimmter Zustand eintritt
`wait on signal_list;`	Warten bis ein bestimmte(s) Signal(e) wechselt
`WAIT FOR time;`	Eine bestimmte Zeit warten
`wait;`	Unbestimmt Warten

Beispiele:

```
Wait until CLK= ‘1‘;
Wait for 10 nS;
Wait on A,B;
```

WAIT Statements

Zürcher Hochschule
für Angewandte Wissenschaften

- WAIT Statements sind sequentielle Statements und dürfen nur im Prozess vorkommen
- Beim Ausführen des WAIT Statements wird der Process unterbrochen und die zugewiesenen Signale werden aktualisiert
- Nach Ausführen der WAIT Bedingung wird der Prozess an der Stelle fortgefahren, wo er unterbrochen wurde
- WAIT ist nicht synthetisierbar

Assert Statement

Zürcher Hochschule
für Angewandte Wissenschaften

```
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.

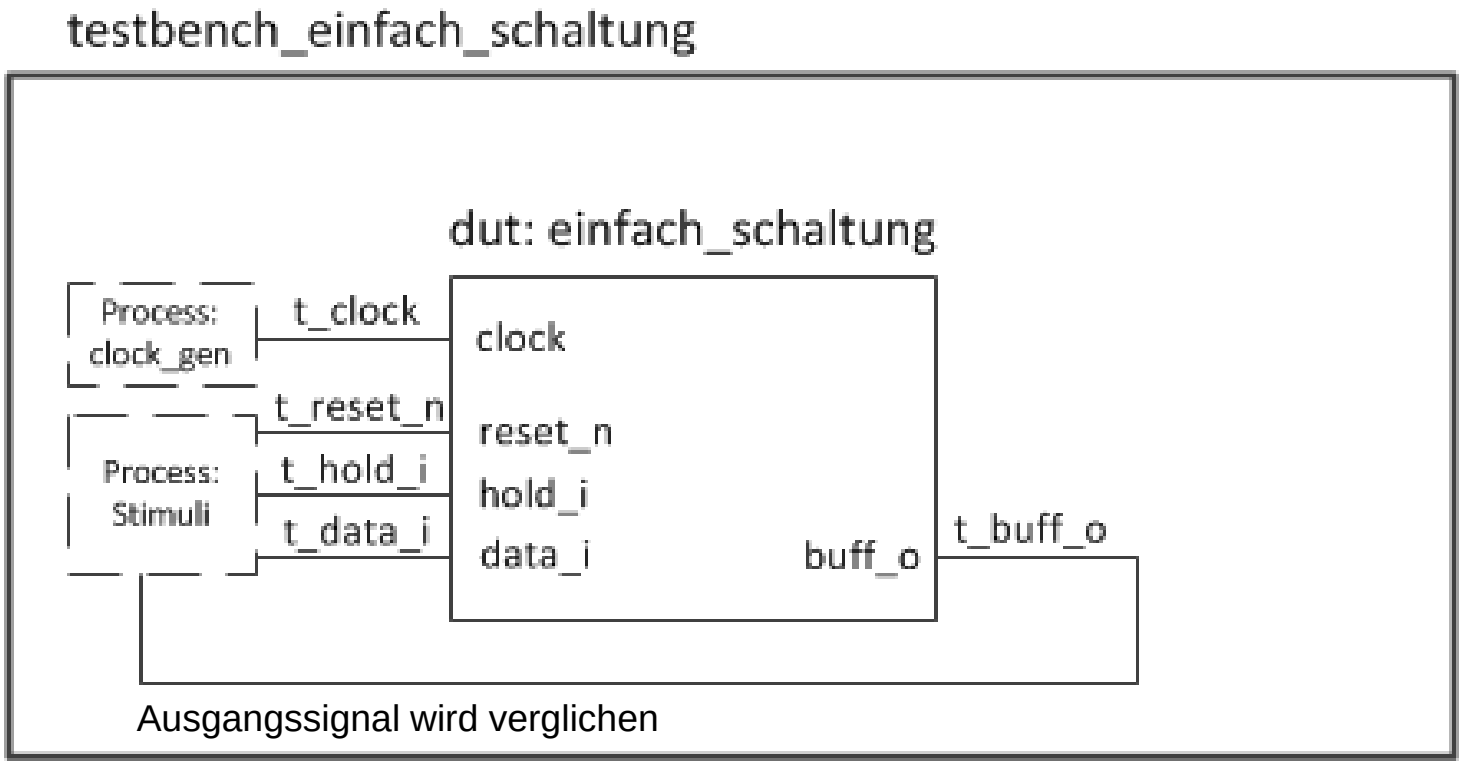
Beispiel: Testprogramm zum Austesten des bcd-gray Kodewandlers

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

Testbench für Lab1- Aufgabe 1



Einfach_Schaltung

Zürcher Hochschule
für Angewandte Wissenschaften

```
-- Library & Use Statements
LIBRARY ieee;
USE ieee.std_logic_1164.all;


-- Entity Declaration
ENTITY einfach_schaltung IS
          PORT (
                    clock     : in std_logic;
                    reset_n : in std_logic;
                    data_i    : in std_logic;
                    hold_i    : in std_logic;
                    buff_o    : out std_logic
                    );
END einfach_schaltung;


-- Architecture Declaration
ARCHITECTURE rtl OF einfach_schaltung IS

          -- Signals & Constants Declaration
SIGNAL buff, next_buff : std_logic ;
```

```
BEGIN
-----------------------------------------
-- Process for combinatorial logic
-----------------------------------------
comb_logic: PROCESS(ALL)
BEGIN
-- hold or update
          IF hold_i='1' THEN
                    next_buff <= buff;
          ELSE
                    next_buff <= data_i;
          END IF;
END PROCESS comb_logic;
-----------------------------------------
-- Process for registers (flip-flops)
-----------------------------------------
flip_flops : PROCESS(clock, reset_n)
BEGIN
          IF reset_n = '0' THEN
          buff <= '0';
          ELSIF RISING_EDGE(clock) THEN
          buff <= next_buff ;
          END IF;
END PROCESS flip_flops;

-----------------------------------------
-- Concurrent Assignements
-- e.g. Assign outputs from
intermediatesignals
-----------------------------------------
buff_o <= buff;
```

Testbench für Lab1- Aufgabe 1

Zürcher Hochschule für Angewandte Wissenschaften

```
-- Testbench-Code: testbench_rsff.vhd
-- History: ...

-- Library & Use Statements
LIBRARY ieee;
USE ieee.std_logic_1164.all;

-- Entity Declaration
ENTITY testbench_einfach_schaltung IS
END testbench_einfach_schaltung;

-- Architecture Declaration
ARCHITECTURE struct OF
testbench_einfach_schaltung IS
 -- Component Declaration
  COMPONENT einfach_schaltung
  PORT (  clock     : in std_logic;
          reset_n : in std_logic;
          data_i    : in std_logic;
          hold_i    : in std_logic;
          buff_o    : out std_logic );
   END COMPONENT einfach_schaltung;
-- Signals & Constants Declaration
 SIGNAL   t_clock   : std_logic;
 SIGNAL   t_reset_n           : std_logic;
 SIGNAL   t_data_i  : std_logic;
 SIGNAL   t_hold_i  : std_logic;
 SIGNAL   t_buff_o  : std_logic;
 CONSTANT  clk_halfp : time := 0.5 us;
```



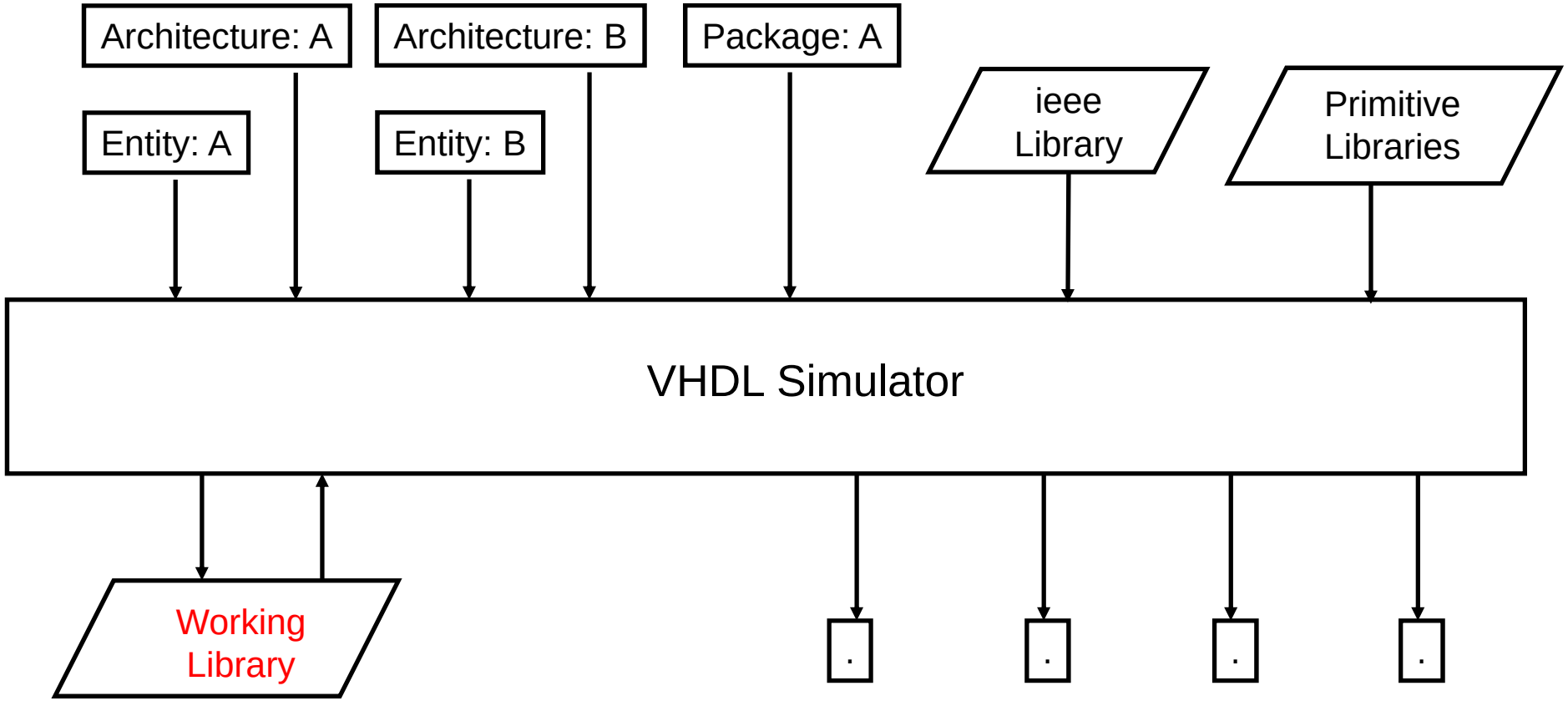
```
-- Begin Architecture
BEGIN
-- Instantiation DUT (Device under Test)
  dut: einfach_schaltung
PORT MAP( clock     => t_clock,
          reset_n => t_reset_n,
          data_i    => t_data_i,
          hold_i    => t_hold_i,
          buff_o    => t_buff_o );

-- Clock Generation Process (with wait)
clock_gen: PROCESS
          BEGIN
          t_clock <= '0';
          WAIT FOR clk_halfp;
          t_clock <= '1';
          WAIT FOR clk_halfp;
END PROCESS clock_gen;
```

Testbench für Lab1- Aufgabe 1

Zürcher Hochschule für Angewandte Wissenschaften

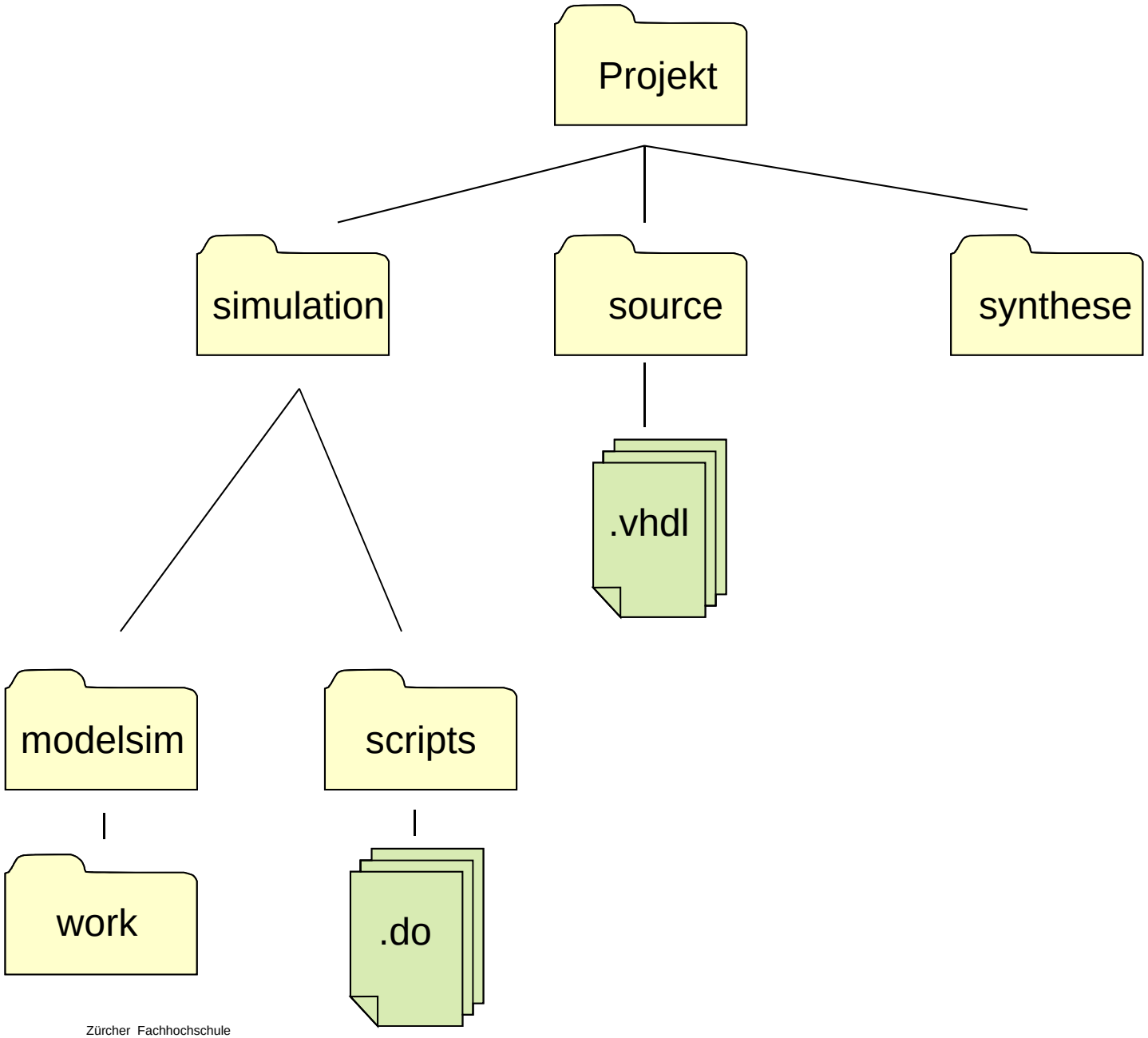
```
-- Stimuli and Check Process (wait & ASSERT)
stimuli: PROCESS
BEGIN
-- initialize all inputs and activate reset_n to initialize the DUT
t_reset_n <= ‘0'; t_data_i <= '1'; t_hold_i <= '0';
WAIT FOR 10*clk_halfp;
-- release reset_n and wait 2 clock-periods
WAIT UNTIL t_clock = '0';
t_reset_n           <= '1';
WAIT FOR  2*clk_halfp;
-- since hold was not active, after clock rising edge check that buff_o = data_i
WAIT UNTIL t_clock = '0';
ASSERT (t_buff_o = t_data_i) REPORT "TEST_1: buff_o not equal data_i" SEVERITY ERROR ;
WAIT FOR  2*clk_halfp;    ...
-- change data_i and check that buff_o follows
WAIT UNTIL t_clock = '0';
t_data_i <= '0';
WAIT FOR 2*clk_halfp;
Assert (t_buff_o = t_data_i) REPORT "TEST_2: buff_o not equal data_i" SEVERITY ERROR ;
-- now set hold and check that buff_o do not -- follow data_i changes
WAIT UNTIL          t_clock = '0';
t_hold_i <= '1'; t_data_i <= '1';
WAIT FOR  2*clk_halfp;
ASSERT (t_buff_o /= t_data_i) REPORT "TEST_3: buff_o equal data_i" SEVERITY ERROR ;

-- stop simulation
WAIT FOR  10*clk_halfp;
ASSERT (FALSE) REPORT "Test programm beendet" SEVERITY FAILURE;
END PROCESS stimuli;
 -- End Architecture
END struct;
```

VHDL Simulator



edaverzeichnis

Zürcher Hochschule
für Angewandte Wissenschaften



Compile Script (compile.do) für funktionale Simulation

Zürcher Hochschule für Angewandte Wissenschaften

```
# create work library

vlib work
```

Bildet Workverzeichniss „work“

```
# compile project files

vcom -2008 -explicit -work work ../../source/testbench_einfach.vhd

vcom -2008 -explicit -work work ../../source/einfach.vhd
```

Compiliert VHDL und legt es im „work“ ab

```
# run the simulation

vsim -t 1ns -lib work work. testbench_einfach
```

Startet Simulator

```
do ../scripts/wave.do
```

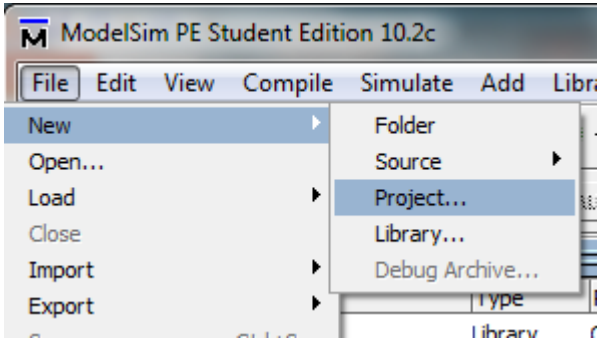
Öffnet Waveform Betrachter

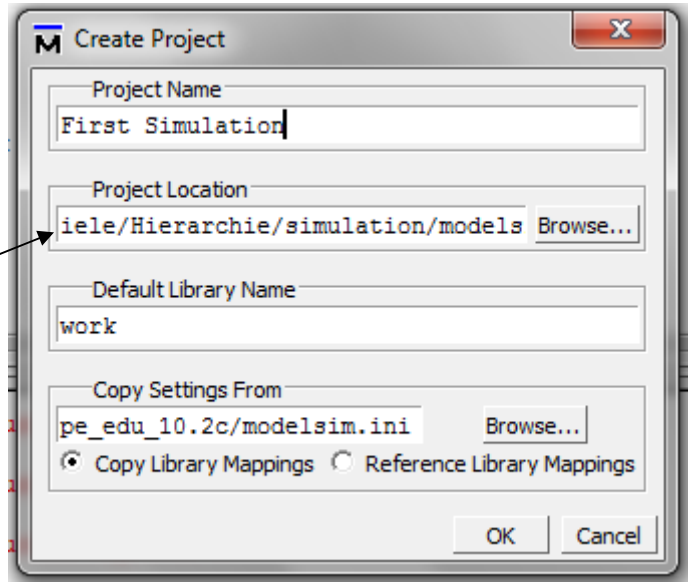
```
run 1800.0 ns
```

Lässt Simulator für 1800 ns laufen

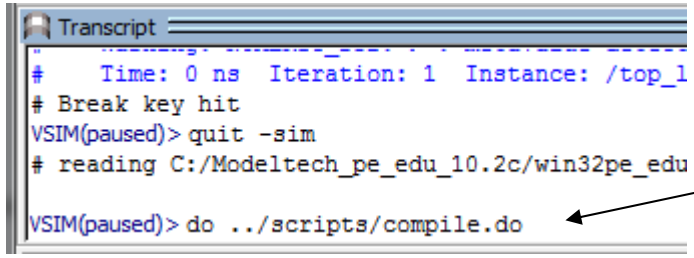
Simulator Starten

Creating a new Project





Arbeitsverzeichnis ins Projektverzeichnis legen



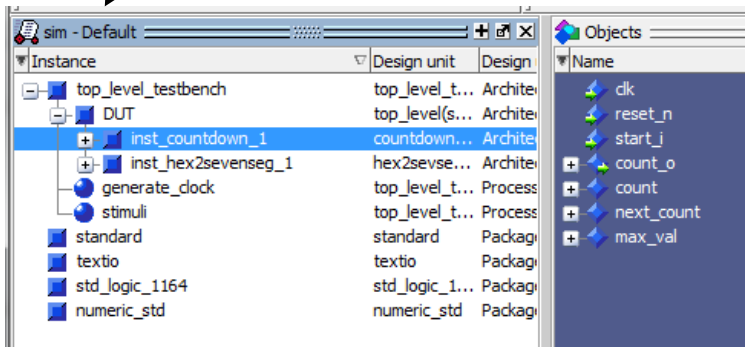
Simulation von Kommandozeile starten

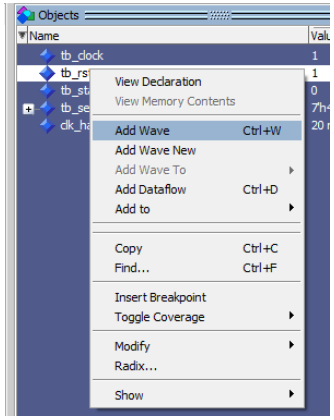
Waveform Window

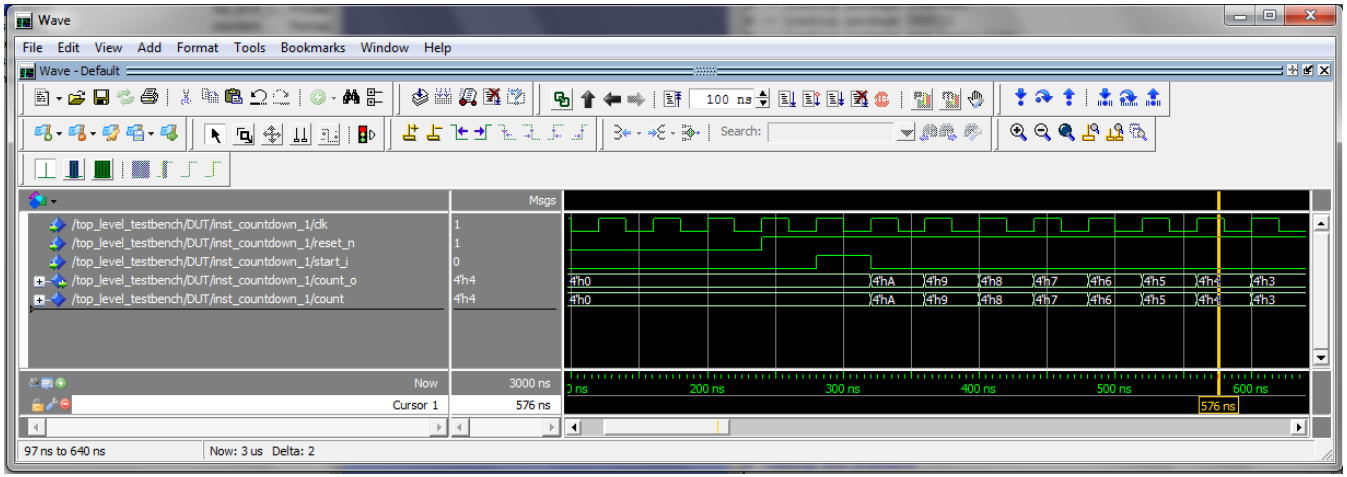
Structure Window
(Design Hierarchie Browser)

Objects Window
(zu simulierende Signale)

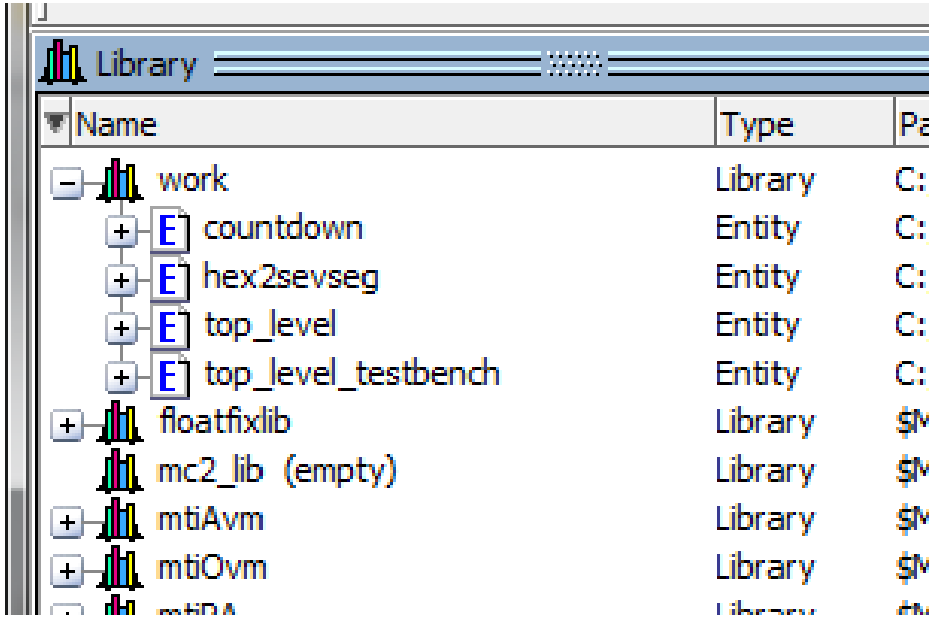
Add Wave







Library Window



VHDL Templates

- Block Design (RTL)
- Block Design (Hierarchical)
- Testbench (Hierarchical & Behavioural)

Template _ Block Design *(RTL)*

```
-- Library & Use Statements

-- Entity Declaration

-- Architecture Declaration
     -- Signals & Constants Declaration

-- Begin Architecture

    ---------------------------------------------
    -- Process for combinatorial logic
    ---------------------------------------------
    -- Process for registers (flip-flops)
    ---------------------------------------------
    -- Concurrent Assignements
    -- e.g. Assign outputs from intermediate signals
    ---------------------------------------------

 -- End Architecture
```

Template _ Block Design *(Hierarchical)*

Zürcher Hochschule
für Angewandte Wissenschaften

```
-- Library & Use Statements
-- Entity Declaration

-- Architecture Declaration
    -- Components Declaration
    -- Signals & Constants Declaration

-- Begin Architecture

    ----------------------------------------
    -- Instantiation Block - 1
    ----------------------------------------
    -- Instantiation Block – 2 ...
    ----------------------------------------
    -- Concurrent Assignments
    -- e.g. Assign outputs from intermediate signals
    ----------------------------------------

 -- End Architecture
```

Template _ Testbench

(Hierarchical & Behavioural)

Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

```
-- Library & Use Statements
-- Entity Declaration

-- Architecture Declaration
    -- Component Declaration
    -- Signals & Constants Declaration

-- Begin Architecture

    ----------------------------------------
    -- Instantiation DUT (Device under Test)
    ----------------------------------------
    -- Clock Generation Process (with wait)
    ----------------------------------------
    -- Stimuli and Check Process (with wait & ASSERT)
    ----------------------------------------

 -- End Architecture
```

nicht synthetisierbare Befehle