DTP2_Vorlesung_05

- Variablen
  - Konzept
  - Sichtbarkeit im Simulator
- Testbench Procedures
  - Procedure für Stimuli und Check

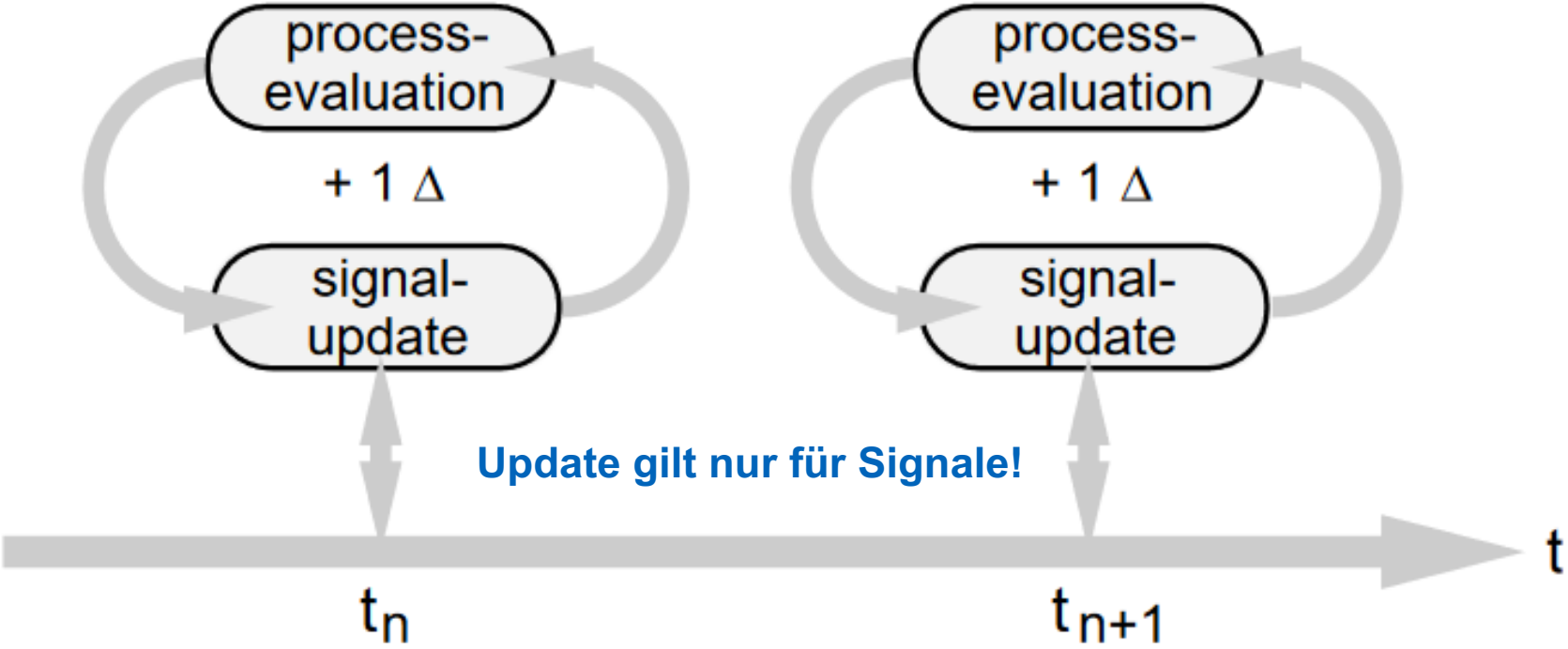
Variablen

Beispiele Prozess:

```
clkgen : process
  begin
    clock
      <= '0';
    wait for 1*clock_period/2;
    clock <= '1';
    wait for 1*clock_period/2;

  end process clkgen;
```

```
comb_logic : process(start_i, count)
  begin

    if (start_i = '1') then
      next_count <= max_val;


    elsif (count > 0) then
      next_count <= count - 1;


    else
      next_count <= count;

    end if;
```

Ablauf von Prozessen



Quelle: Lehman, Wunder, Selz: «Schaltungsdesign mit VHDL»

Syntax der Variablen

Zürcher Hochschule für Angewandte Wissenschaften

Variablen kann man benutzen, um innerhalb eines Prozesses oder einer Function oder Procedure einen Wert temporär zu speichern

```
VARIABLE variablen_name : type [:= initialisierung];
```

Variablenzuweisung:

```
''Variablenname'' := ''Wert'';
```

Zuweisungsoperator

Beispiel:

```
VARIABLE summe : std_logic_vector (2 downto 0) := "101" ;
```

Verwendung einer Variablen

Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

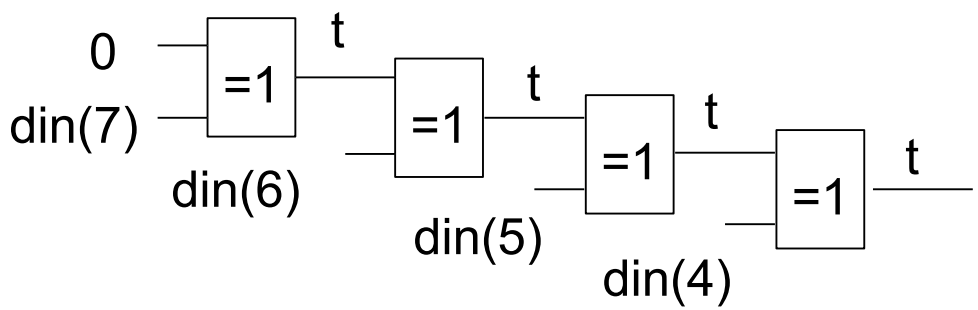
Beispiel 1

```
architecture comb of voter is
begin

  comb_voter : process (all) is

    variable x : std_logic_vector(2 downto 0) := "000";

  begin  -- process comb_voter
    x := a & b & c;
    case x is
      when "110"  => y <= '1';
      when "101"  => y <= '1';
      when "011"  => y <= '1';
      when "111"  => y <= '1';
      when others => y <= '0';
    end case;
  end process comb_voter;
end architecture comb;
```

Verwendung einer Variablen

Beispiel 2

```
architecture comb of parity_generator is

  signal din : std_logic_vector(7 downto 0);
  signal parity_out : std_logic;

begin

  parity : process (all) is
    variable t : std_logic := '0';
  begin
    for i in 7 downto 0 loop
      t := t xor din(i);
    end loop;  -- i
    parity_out <= t;
  end process parity;
end architecture comb;
```



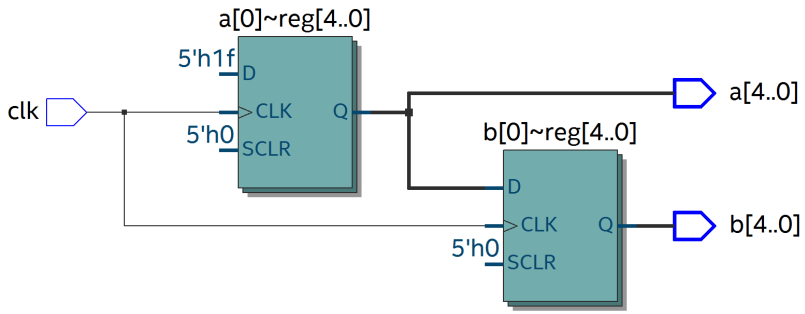
Variablen

- Kann man nur in Prozessen oder Unterprogrammen benutzen.
- Werden deshalb zwischen den Zeilen „Prozess“ und „Begin“ definiert.
- Variablen müssen vor Gebrauch initialisiert werden.
- Sind Global nicht verfügbar (ausserhalb des Prozess)
- Will man Variablen ausserhalb eines Prozesses weiterbenutzen, muss ein Signal zugewiesen werden. sig <= var;
- Sind nach der Zuweisung sofort gültig, nicht erst beim Verlassen des Prozesses.
- Können als Index oder temporärer Speicher innerhalb eines Prozesses benutzt werden.
- Zuweisung durch :=

Variable vs. Signal

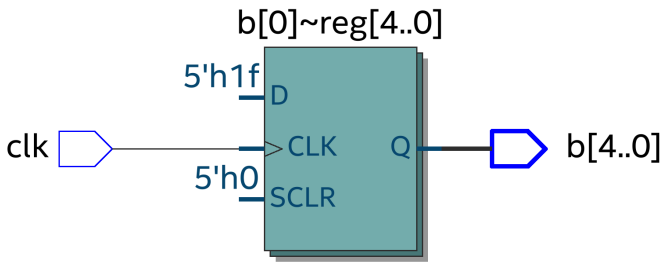
Variante 1:

```
signal a,b : std_logic_vector(4 downto 0);

process (CLK)
    begin
        if (rising_edge(clk)) then
            a <= "11111";
            b <= a;
        end if;
end process;
```



Variante 2:

```
process (CLK)
   variable a : std_logic_vector(4 downto 0);
 begin
        if (rising_edge(clk)) then
            a := '11111';
            b <= a;
        end if;
end process;
```



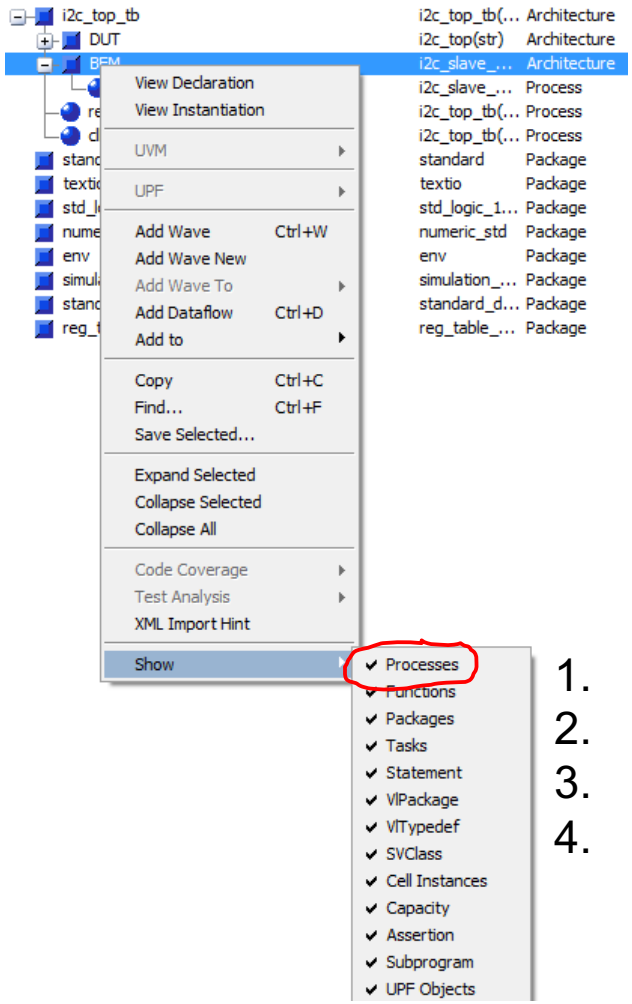
Signale vs. Variablen

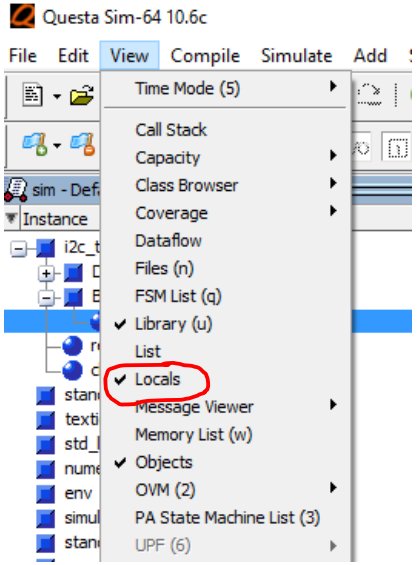
Zürcher Hochschule
für Angewandte Wissenschaften

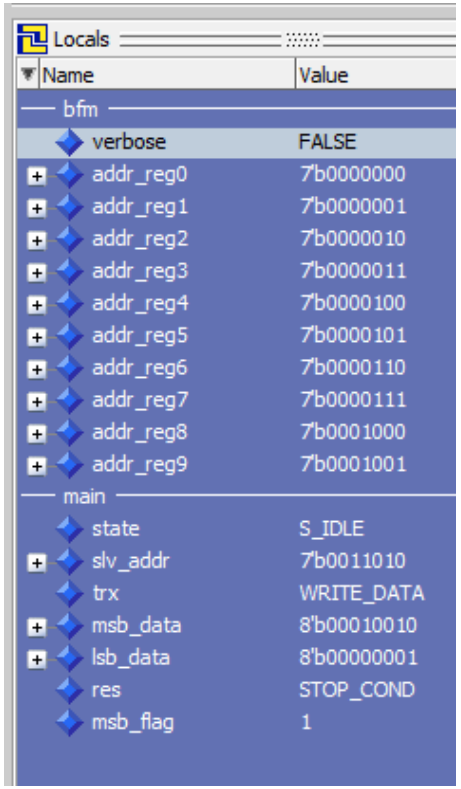
zh
aw

Signale	Variablen
Existieren global in der gesamten Architektur	Nur lokal im Prozess lesbar
Werden am Ende des Prozesses aktualisiert.	Werden immer sofort aktualisiert
Werden zwischen „Architectur“ und „Begin“ deklariert	Werden zwischen „Process“ und „Begin“ deklariert
Zuweisung v <= ‘1‘	Zuweisung v := ‘1‘
Erzeugt ein FF nach rising_edge(clk)	Erzeugt <u>kein</u> FF nach rising_edge(clk)

Variablen im Simulator anzeigen lassen



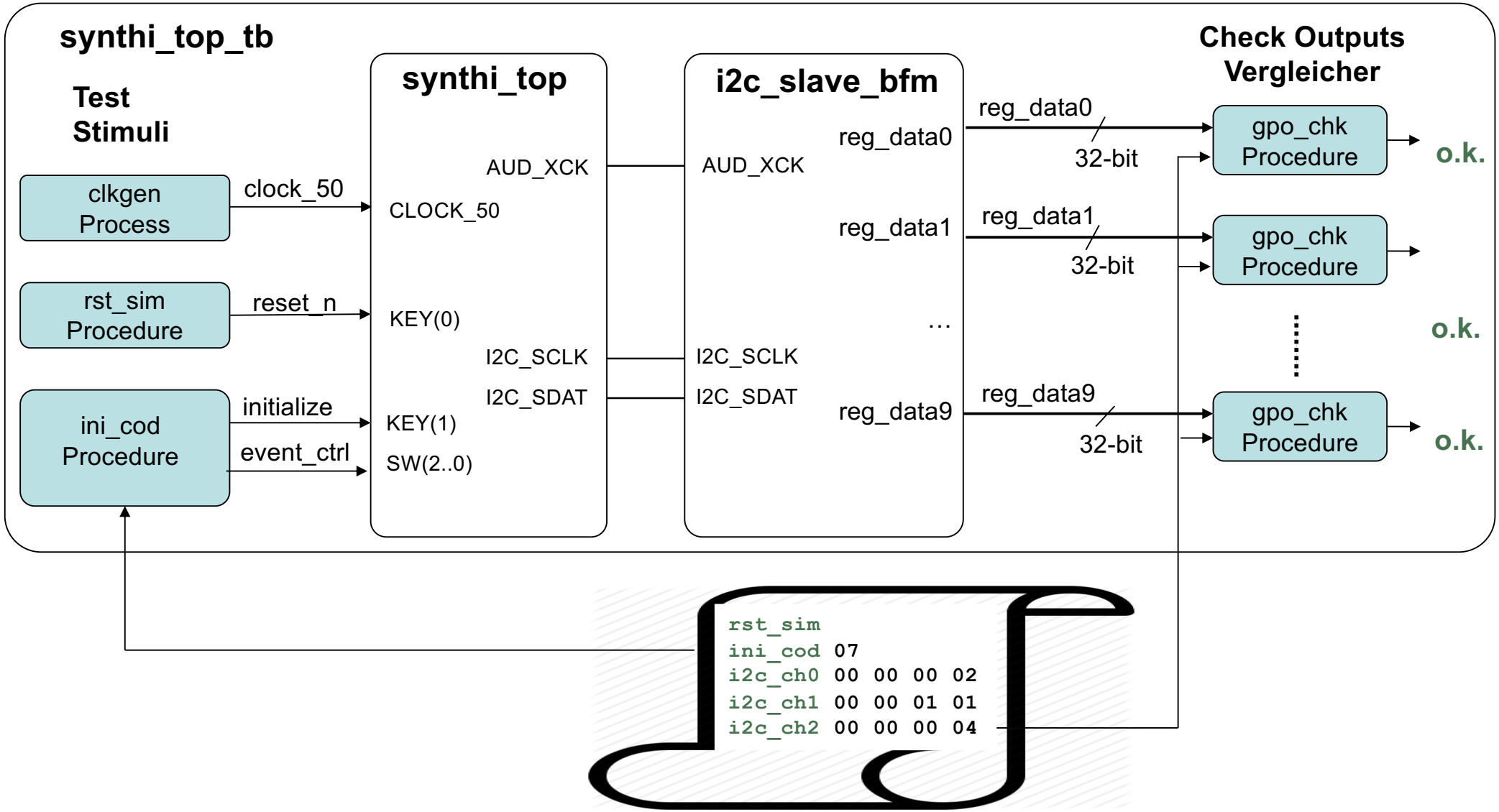




1. Select Entity
2. Rechte Maustaste → Show →Processes
3. Select Process
4. View → Locals

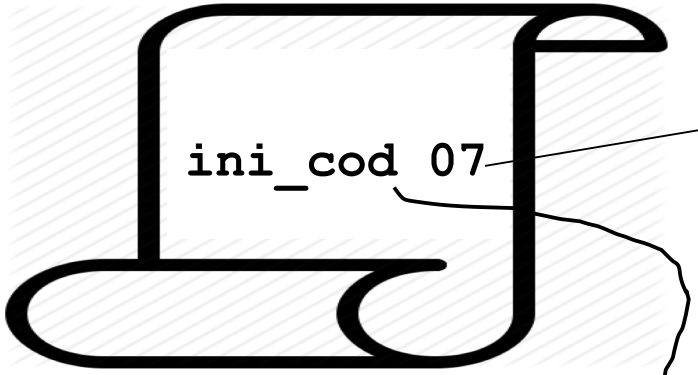
Testbench Procedures

Testbench Ausbau für i2c



Test Script: testcase.dat

Stimulating the Codec Controller



Bits (2..0) von Argument 1 definieren event_ctrl

Test Vector Script: testcase.dat

synthi_top_tb

```
ini_cod(tv, SW(2 downto 0), KEY(1));
```

synthi_top

codec_controller

KEY(1) — initialize event_ctrl

SW(2 downto 0) — write_o, write_data

Record *test_vect*, used with variable *tv*

```
type test_vect is record
    arg1          : std_logic_vector(7 downto 0);
    arg2          : std_logic_vector(7 downto 0);
    arg3          : std_logic_vector(7 downto 0);
    arg4          : std_logic_vector(7 downto 0);
    obs_data      : std_logic_vector(31 downto 0);
    fail_flag     : boolean;
    clock_period  : time;
```

Definiert in simulation_pkg.vhd

Procedure ini_cod in standard_driver_pkg.vhd

```
procedure ini_cod
  (
    variable tv              : inout test_vect;
    signal codec_mode        : out   std_logic_vector(2 downto 0);
    signal codec_initialize  : out   std_logic
    ) is

  variable line_out : line;

begin
  codec_initialize <= '1';
  codec_mode       <= tv.arg1(2 downto 0);
  wait for 8*tv.clock_period;

  codec_initialize <= '0';
  wait for 4*tv.clock_period;
  codec_initialize <= '1';

  wait for 48000 * tv.clock_period;

end procedure ini_cod
```

Checking the Output of the I2C_slave_bfm

```
i2c_ch0 00 00 01 4a
i2c_ch1 00 00 00 f3
i2c_ch3 00 00 00 ff
…
```

Vergleichswert = 4 x Byte = 32-bit von denen nur die 9 LSb verwendet werden

Test Vector Script: testcase.dat

synthi_top_tb

```
elsif cmd = string'("i2c_ch0") then
    gpo_chk(tv, reg_data0);
elsif cmd = string'("i2c_ch1") then
    gpo_chk(tv, reg_data1);
```

I2c_slave_bfm

I2C_SCLK
I2C_SDAT

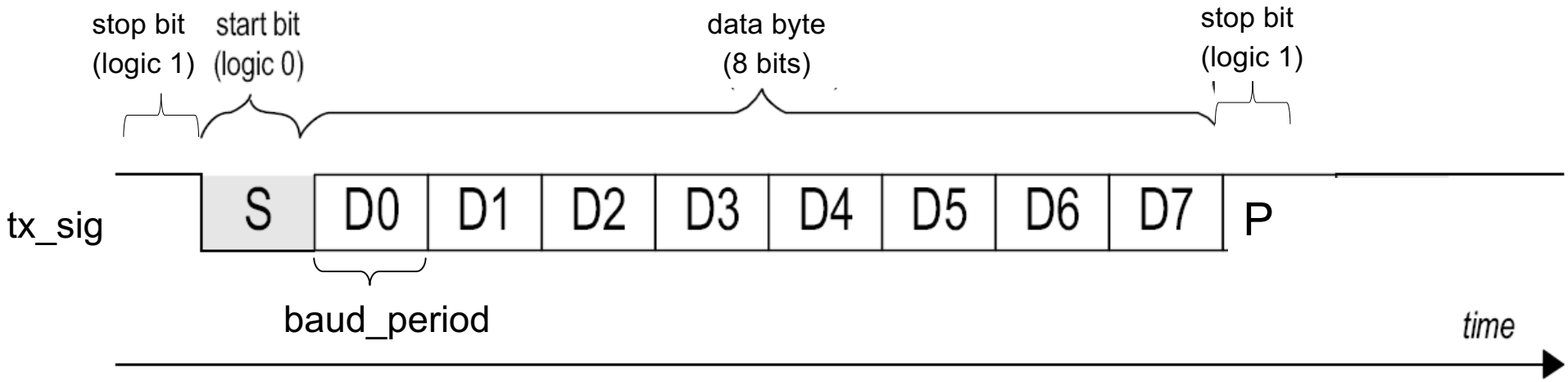
reg_data0
reg_data1
reg_data2
…
reg_data9

```
signal reg_data0 : std_logic_vector(31 downto 0);
signal reg_data1 : std_logic_vector(31 downto 0);
signal reg_data2 : std_logic_vector(31 downto 0);
```

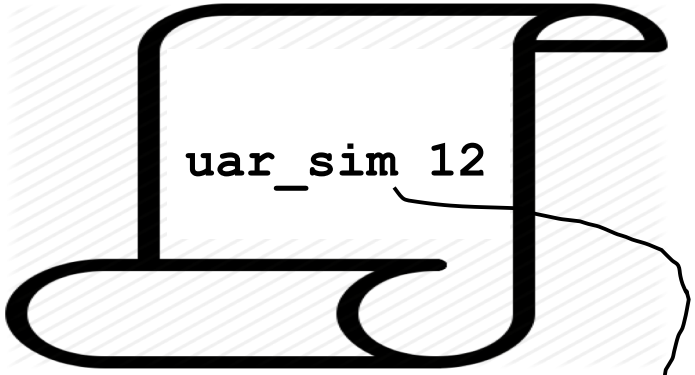
Procedure gpo_chk

```
procedure gpo_chk
   ( variable tv    : inout test_vect;
     signal in_sig : in    std_logic_vector(31 downto 0)
     ) is

   variable line_out : line;           -- Line buffers

 begin
   if tv.arg1 = in_sig(31 downto 24) and
      tv.arg2 = in_sig(23 downto 16) and
      tv.arg3 = in_sig(15 downto 8)  and
      tv.arg4 = in_sig(7 downto 0)

   then
     tv.fail_flag := false;
     hwrite(line_out, tv.arg1);
     hwrite(line_out, tv.arg2);
     …
     write(line_out, string'(" RETURNED FROM DUT, O.K. "));
     writeline(OUTPUT, line_out);      -- write the message
   else
     tv.fail_flag := true;
     write(line_out, string'("ERROR: COMPARE FAILED. SIGNALS RETURNED "));
     hwrite(line_out, in_sig);
     write(line_out, string'(" SIGNALS EXPECTED "));
     hwrite(line_out, tv.arg1);
     hwrite(line_out, tv.arg2);
     …
     writeline(OUTPUT, line_out);
   end if;

 end gpo_chk;
```

Beispiel anderer Procedures: Z.B. definition des UART Signales



Beispiel: UART Stimulus



Test Vector Script: testcase.dat

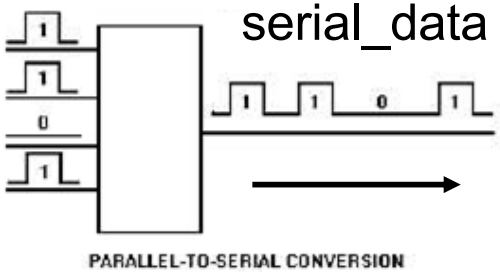
VHDL Record mit Kommando und Argumenten

```
uar_sim(tv, serial_data_pin);
```

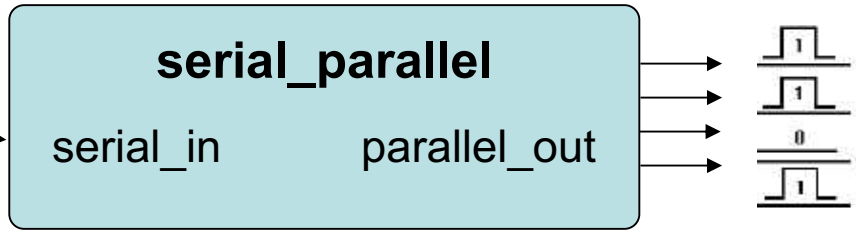
```
signal serial_data : std_logic;
```

Procedure: uar_sim

Record der Zeile (tv)



DUT



Procedure uar_sim

```
procedure uar_sim
  ( variable tv    : inout test_vect;
    signal tx_sig : out   std_logic ) is

  variable line_out    : line;        -- Line buffers
  variable baud_period : time
  variable stop_bit    : std_logic := '0';
begin
  if tv.arg2 = std_logic_vector(to_unsigned(1, 8)) then
    baud_period := baud_period_31_250;
  else
    baud_period := baud_period_115_200_000;
  end if;
  if tv.arg3 = std_logic_vector(to_unsigned(1, 8)) then
    stop_bit := '0'; else  stop_bit := '1';  end if;

  tx_sig <= '1';                       -- Before start-bit: set tx signal to one
  wait for 2* baud_period;             -- Wait long enough between transmissions
  tx_sig <= '0';                       -- start-bit
  wait for baud_period;                -- Apply start bit for one baud period
  txloop : for i in 0 to 7 loop        -- Transmit eight bits
             tx_sig <= tv.arg1(i);
             wait for baud_period;
  end loop txloop;
  tx_sig <= stop_bit;                 -- This is the stop bit
  wait for baud_period;
  tx_sig <= '1';                       -- stop-bit
  wait for baud_period;
```

Audio Codec Register Map

Register Addresses (7-bits) — B15 … B9

Register Data (9-bits) — B8 … B0

REGISTER	B 15	B 14	B 13	B 12	B 11	B 10	B 9	B8	B7	B6	B5	B4	B3	B2	B1	B0
R0 (00h)	0	0	0	0	0	0	0	LRIN BOTH	LIN MUTE	0	0	LINVOL				
R1 (02h)	0	0	0	0	0	0	1	RLIN BOTH	RIN MUTE	0	0	RINVOL				
R2 (02h)	0	0	0	0	0	1	0	LRHP BOTH	LZCEN	LHPVOL						
R3 (06h)	0	0	0	0	0	1	1	RLHP BOTH	RZCEN	RHPVOL						
R4 (08h)	0	0	0	0	1	0	0	0	SIDEATT		SIDETONE	DAC SEL	BY PASS	INSEL	MUTE MIC	MIC BOOST
R5 (0Ah)	0	0	0	0	1	0	1	0	0	0	0	HPOR	DAC MU	DEEMPH		ADC HPD
R6 (0Ch)	0	0	0	0	1	1	0	0	PWR OFF	CLK OUTPD	OSCPD	OUTPD	DACPD	ADCPD	MICPD	LINEINPD
R7 (0Eh)	0	0	0	0	1	1	1	0	BCLK INV	MS	LR SWAP	LRP	IWL		FORMAT	
R8 (10h)	0	0	0	1	0	0	0	0	CLKO DIV2	CLKI DIV2	SR				BOSR	USB/ NORM
R9 (12h)	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	ACTIVE
R15(1Eh)	0	0	0	1	1	1	1	RESET								
	ADDRESS							DATA								

Audio Codec Settings

Example: W8731_BYPASS_M12DB_8K

Zürcher Hochschule für Angewandte Wissenschaften

Register	Address Bits B[15:9]	Data Bits B[8:0]	Beschreibung
R0 LEFT_LINE_IN	*"0000000" = 0x00*	*"000001111" = 0x00F*	No mute, no both, volume -12 dB
R1 RIGHT_LINE_IN	*"0000001"* = 0x01	*"000001111" = 0x00F*	No mute, no both, volume -12 dB
R2 LEFT_HP_OUT	*"0000010"* = 0x02	"001101101" = 0x06D	No zero-cross, no both, volume -12dB
R3 RIGHT_HP_OUT	*"0000011"* = 0x03	"001101101" = 0x06D	No zero-cross, no both, volume -12dB
R4 ANALOG_AP	*"0000100"* = 0x04	"000001010" = 0x00A	Mute microphone, disable DAC, enable bypass, disable sidetone,…
R5 DIGITAL_AP	*"0000101"* = 0x05	"000001000" = 0x008	Enable DAC mute, disable de-emphasis, …
R6 POWER_DOWN	*"0000110"* = 0x06	"000000000" = 0x000	No power down mode selected
R7 DIGITAL_AI	*"0000111"* = 0x07	"000000010" = 0x002	Audio-IF I2S Mode, audio data 16 bits wide, …
R8 SAMPLING	*"0001000"* = 0x08	"000001100" = 0x00C	Normal mode, MCLK 12,28MHz, sample rate 48kHz (both ADC and DAC), SR=[0011], …
R9 DIGITAL_ACTIVATE	*"0000111"* = 0x09	"000000001" = 0x001	Activate Audio Digital Interface
~~*R10 – R14 RESERVED A-E*~~	~~0x0A - 0x0E~~	~~"000000000" = 0x000~~	~~No need to write~~
~~*R15 RESET*~~	~~*"0001111"* = 0x0F~~	~~"000000000" = 0x000~~	~~Only write if SW-reset required~~

reg_table_pkg.vhd

```
-- C_ADDR_LEFT/RIGHT_LINE_IN
constant C_LINE_IN_LINVOL_P12DB : std_logic_vector(8 downto 0) := "000011111";  -- Line input volume +12 dB
constant C_LINE_IN_LINVOL_0DB   : std_logic_vector(8 downto 0) := "000010111";  -- Line input volume 0 dB
constant C_LINE_IN_LINVOL_M12DB : std_logic_vector(8 downto 0) := "000001111";  -- Line input volume -12 dB
constant C_HP_OUT_HPVOL_P6DB  : std_logic_vector(8 downto 0) := "001111111";  -- Headphone volume +6dB
constant C_HP_OUT_HPVOL_P4DB  : std_logic_vector(8 downto 0) := "001111101";  -- Headphone volume +4dB
constant C_HP_OUT_HPVOL_P2DB  : std_logic_vector(8 downto 0) := "001111011";  -- Headphone volume +2dB
constant C_HP_OUT_HPVOL_0DB   : std_logic_vector(8 downto 0) := "001111001";  -- Headphone volume 0dB
constant C_HP_OUT_HPVOL_M2DB  : std_logic_vector(8 downto 0) := "001110111";  -- Headphone volume -2dB
constant C_HP_OUT_HPVOL_M4DB  : std_logic_vector(8 downto 0) := "001110101";  -- Headphone volume -4dB
constant C_HP_OUT_HPVOL_M12DB : std_logic_vector(8 downto 0) := "001101101";  -- Headphone volume -12db.
constant C_HP_OUT_HPVOL_MUTE  : std_logic_vector(8 downto 0) := "000000000";  -- Headphone volume MUTE
constant C_ANALOG_AP_DACSEL  : std_logic_vector(8 downto 0) := (4 => '1', others => '0');  -- Add DAC signal to output
constant C_ANALOG_AP_BYPASS  : std_logic_vector(8 downto 0) := (3 => '1', others => '0');  -- Bypass switch
constant C_ANALOG_AP_MUTEMIC : std_logic_vector(8 downto 0) := (1 => '1', others => '0');  -- Microphone mute.
```

reg_table_pkg.vhd

```
constant C_W8731_ANALOG_MUTE_LEFT : t_codec_register_array := (
      000001111 = 00f
0 => "000000000" or C_LINE_IN_LINVOL_M12DB,
      000001111 = 00f
1 => "000000000" or C_LINE_IN_LINVOL_M12DB,

2 => "000000000" or C_HP_OUT_HPVOL_MUTE,           -- Mute   left

3 => "000000000" or C_HP_OUT_HPVOL_0DB,            -- louder  right
      000001010 = 00a
4 => "000000000" or C_ANALOG_AP_MUTEMIC or C_ANALOG_AP_BYPASS,
      000001000 = 008
5 => "000000000" or C_DIGITAL_AP_DACMU or C_DIGITAL_AP_DEEMP_DIS,

6 => "000000000" or C_POWER_DOWN_NONE,
      000000010 = 002
7 => "000000000" or C_DIGITAL_AI_IWL_16 or C_DIGITAL_AI_FORMAT_I2S,

8 => "000000000",                                  -- 48kHz with mclk=12.288MHz

9 => "000000000" or C_DIGITAL_ACTIVATE_ACTIVE);
```

Revision Control System

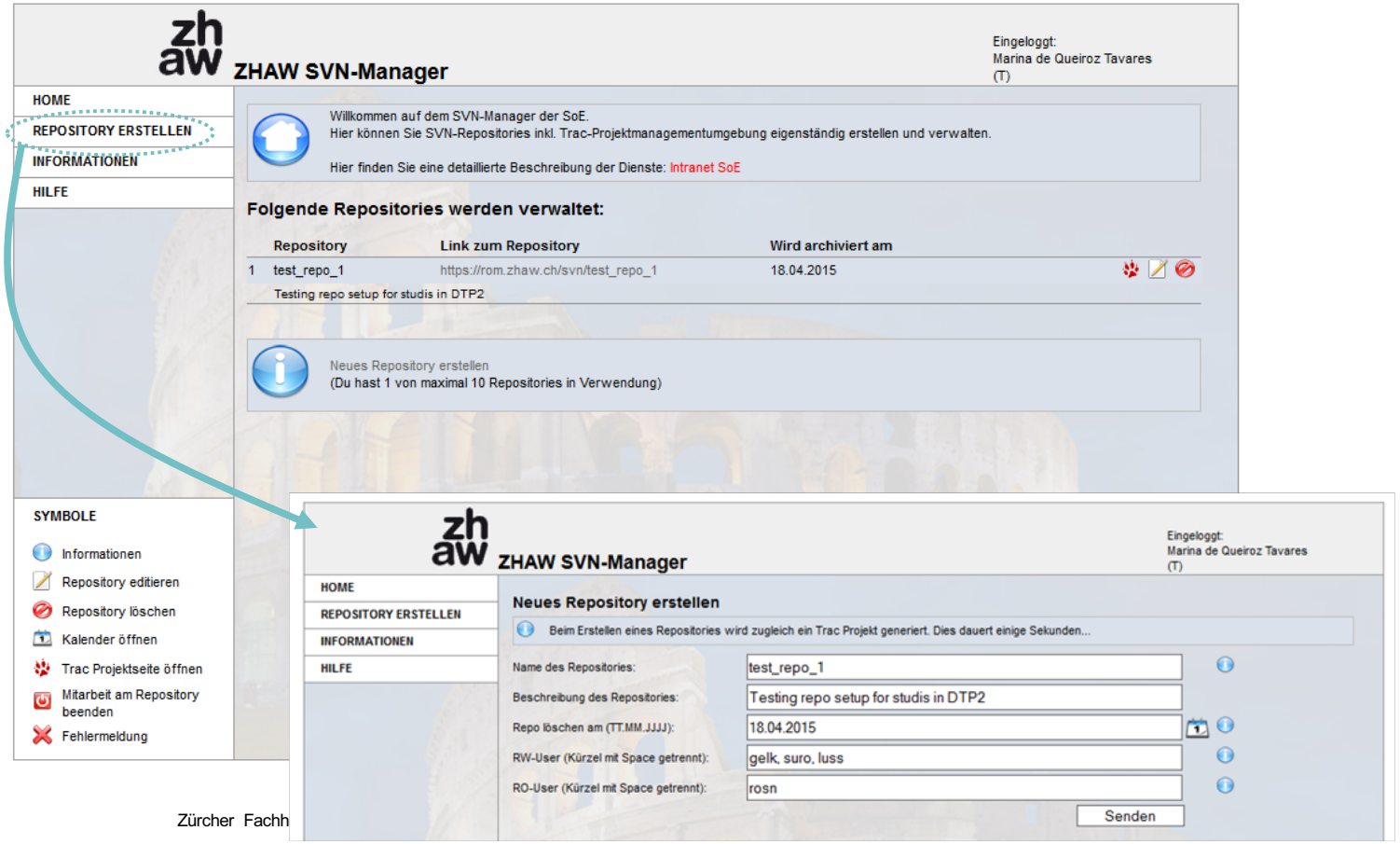
Version Management Tool (SVN - Subversion)

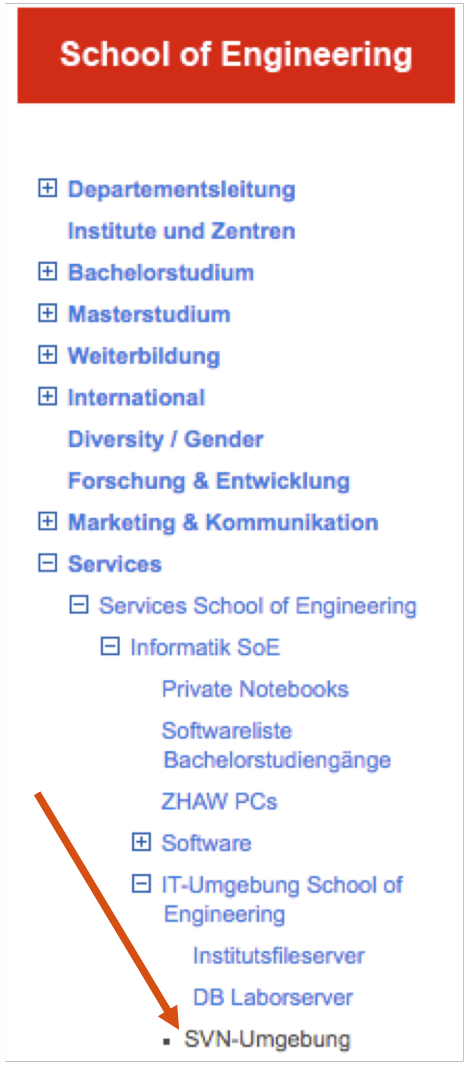
Opening a Project => SVN Server

Managing Project Data => SVN Client (Tortoise)

SVN-Umbegung & SVN-Server für Studierende

https://rom.zhaw.ch/oder link unten





SVN Projekt Beispiel
step-by-step

(A) SVN Projekt erstellen: auf den SVN-Server!

Neues Repository erstellen

Beim Erstellen eines Repositories wird zugleich ein Trac Projekt generiert. Dies dauert einige Sekunden...

Name des Repositories:	dt1-test
Beschreibung des Repositories:	Dateien fue DT1 Praktikum
Repo archivieren am (TT.MM.JJJJ):	04.08.2011
RW-User (Kürzel mit Space getrennt):	dqtm
RO-User (Kürzel mit Space getrennt):	

Senden

SVN Projekt Beispiel
step-by-step

Projekt Daten Verwalten mit SVN Client

(B) Client Tool herunterladen und installieren

e.g. Tortoise

Download: http://tortoisesvn.net

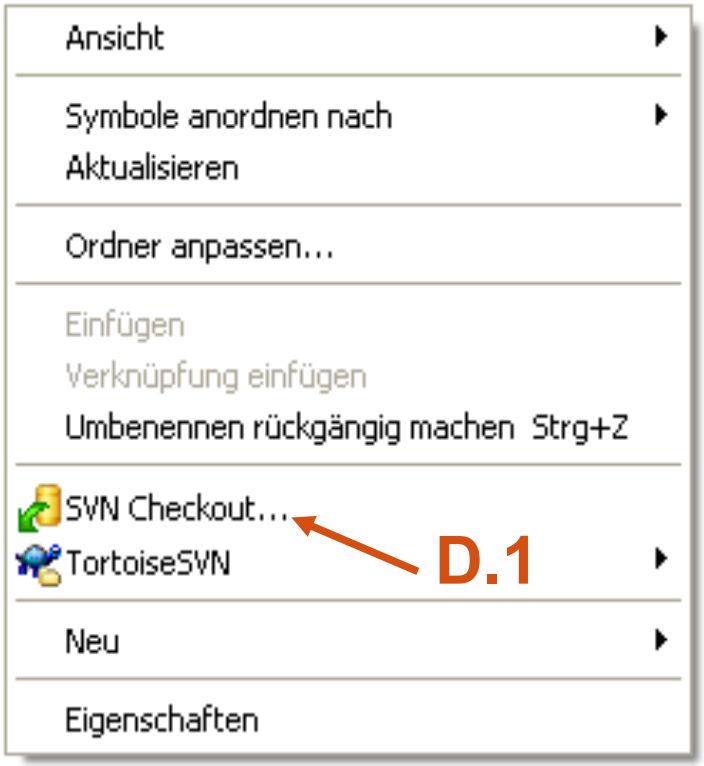
(C) Neues Verzeichnis in eigenes PC erstellen für lokale Kopie von Repository

BMK: weitere Dokus über SVN

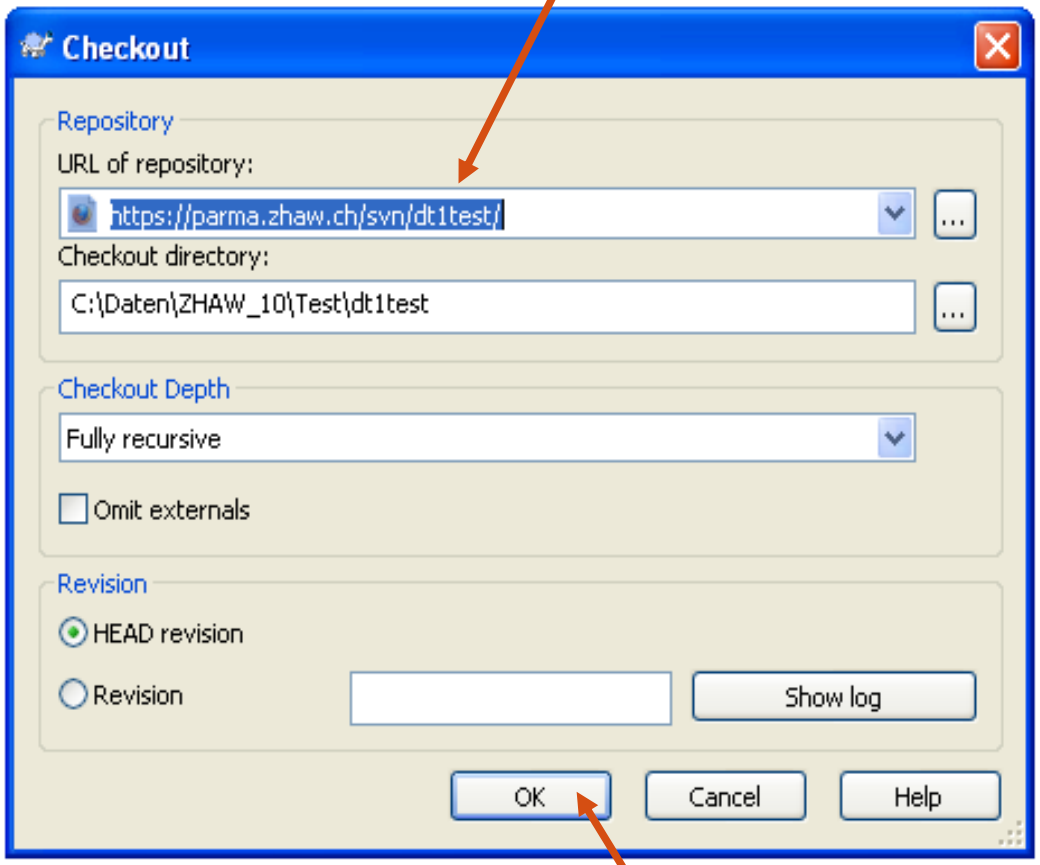
- https://intra.zhaw.ch/fileadmin/user_upload/80_Dept/T_School_of_Engineering/_Dokumente/Services/Informatik_SoE/IT-Umgebung/Subversion-TRAC_ReadMe.pdf
- http://subversion.apache.org/

SVN Projekt Beispiel
step-by-step

(D) Checkout: lokale Kopie von Repository



D.2 URL kopieren

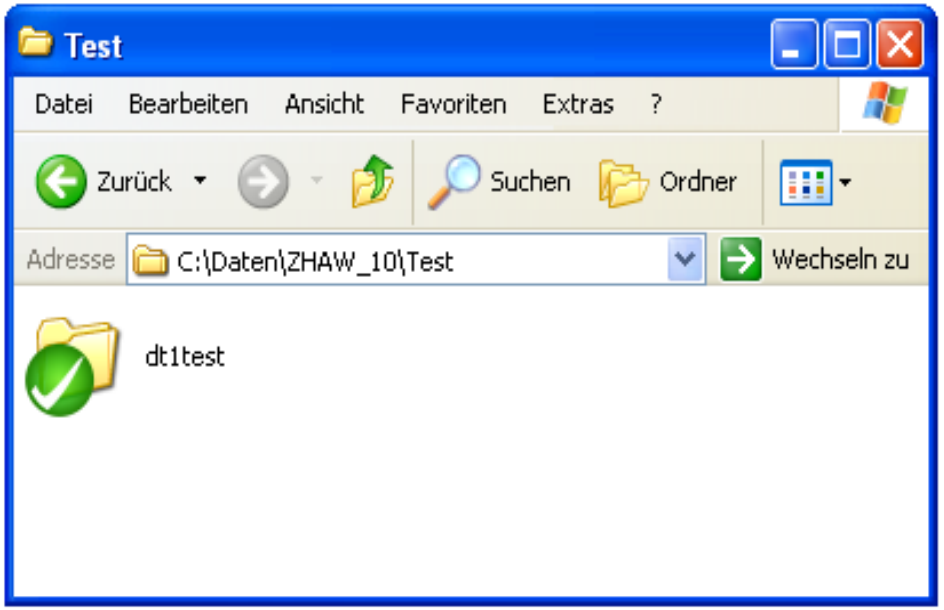


D.3

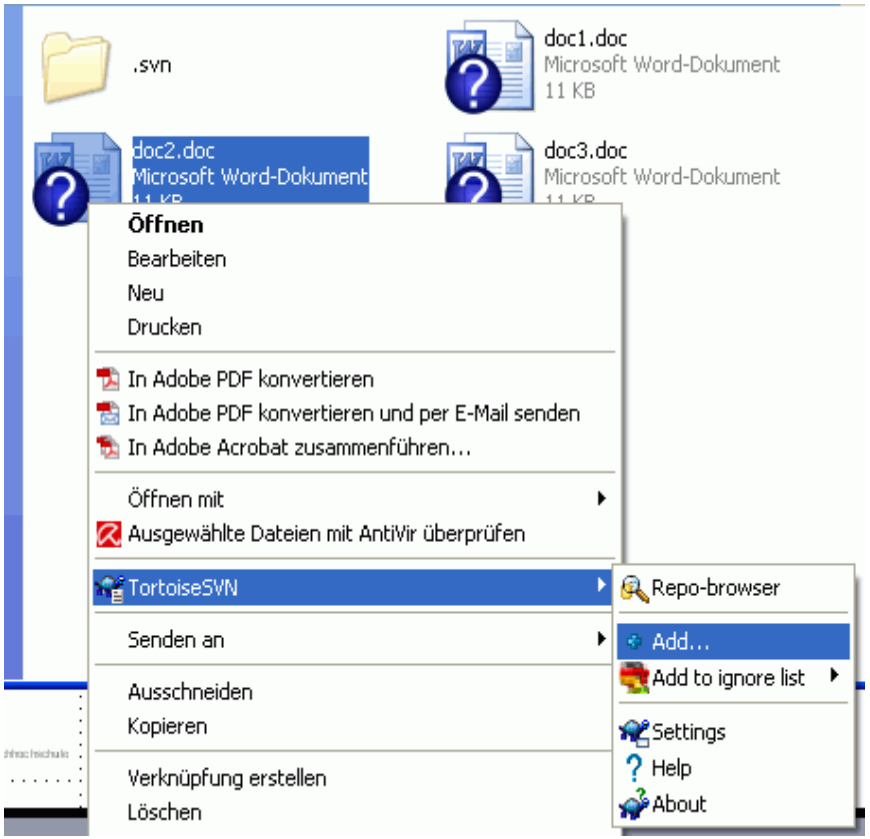
Projekt Daten Verwalten mit Tortoise:
Neue Dateien im Repository anlegen

Zürcher Hochschule
für Angewandte Wissenschaften

1. Dateien in das entstandene Verzeichnis kopieren



2. Neue Dateien zu Repository hinzufügen



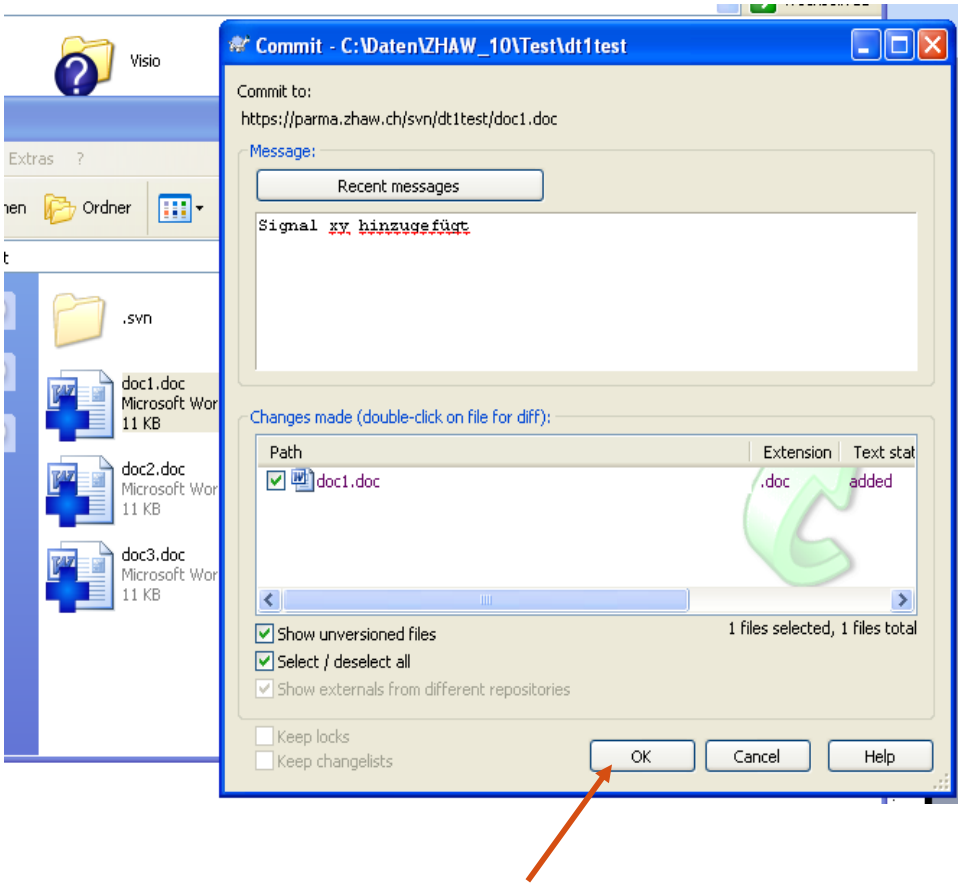
Projekt Daten Verwalten mit Tortoise:
Check-In einer Datei

Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

1. Commit aufrufen

2. Check-In Notiz hinzufügen

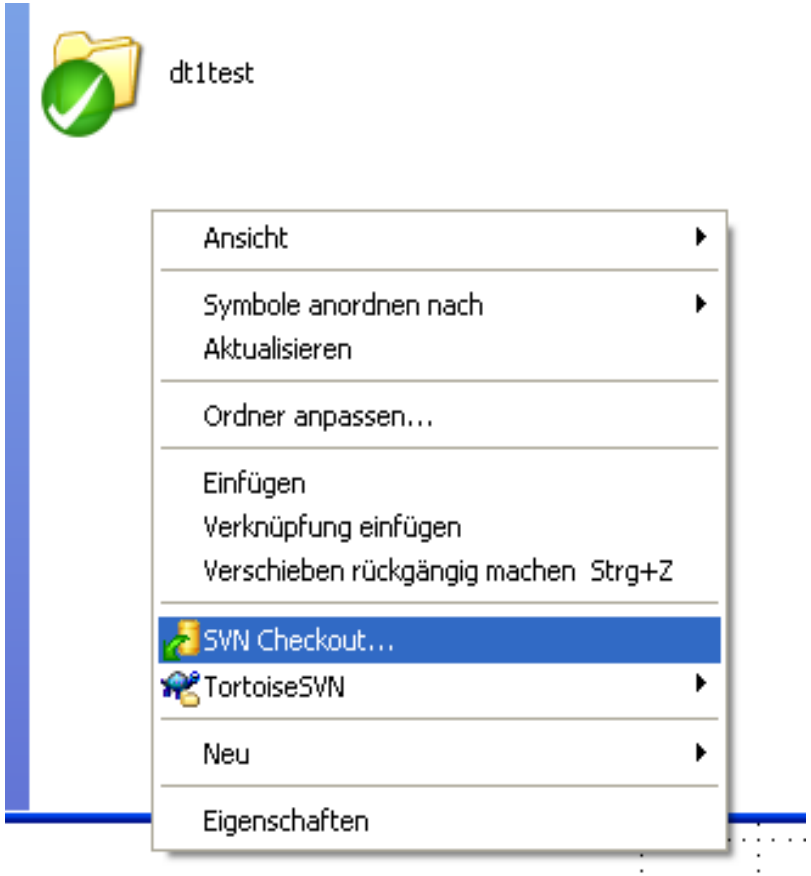


Projekt Daten Verwalten mit Tortoise:
Ein ganzes Repository auschecken

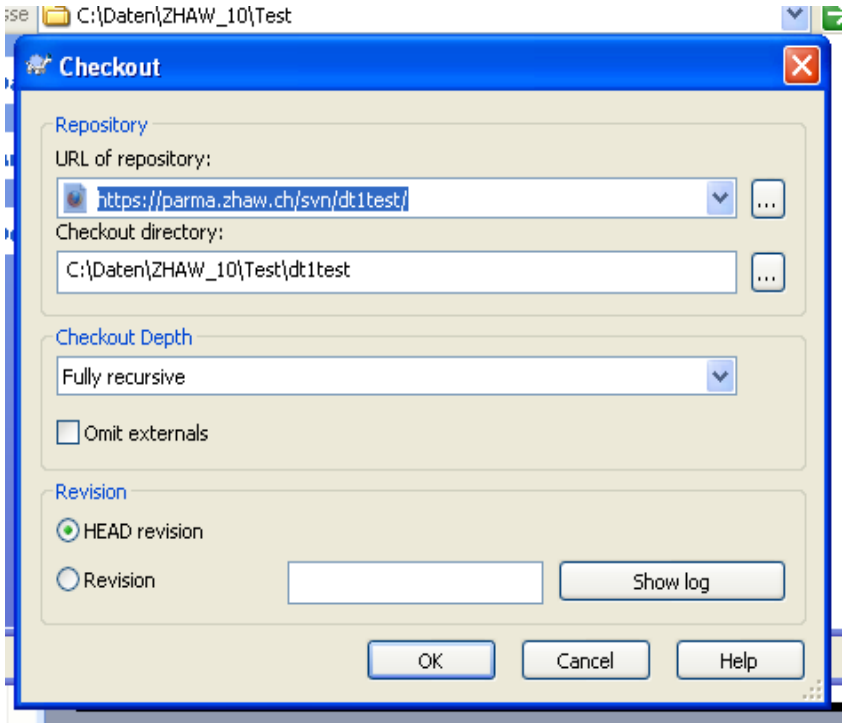
Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

1. Check-Out aufrufen

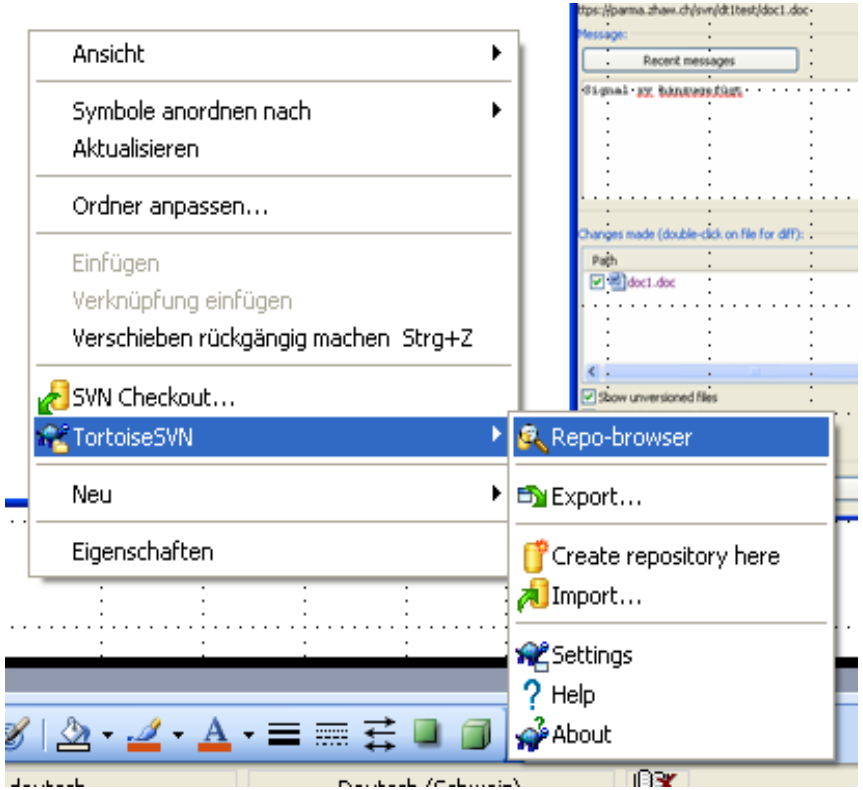


2. URL eingeben

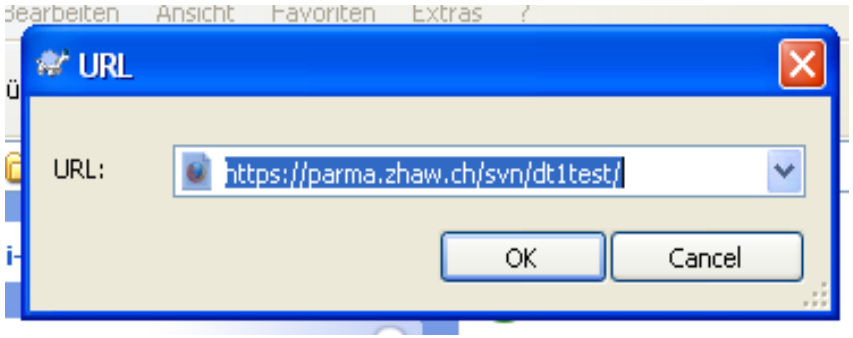


Projekt Daten Verwalten mit Tortoise:
Ein bestimmtes Verzeichnis aus einem Repository auschecken (1)

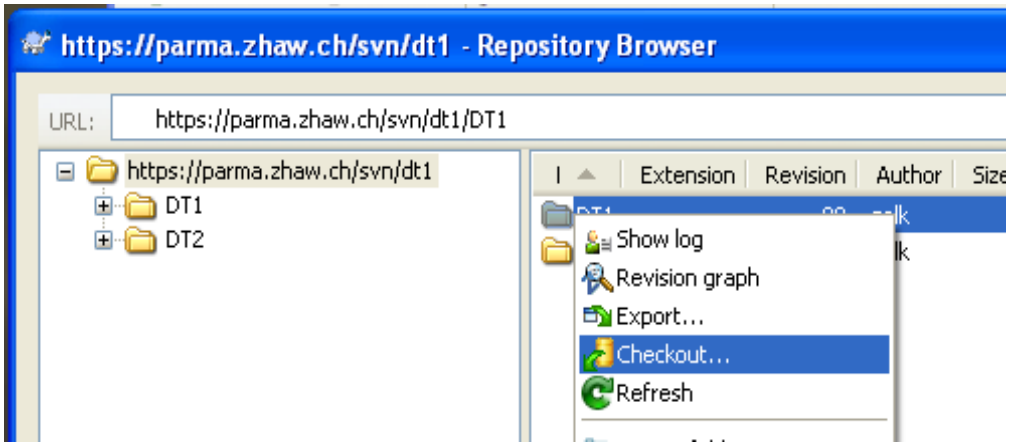
1. Den Repo Browser aufrufen



2. Die URL des Repositories eingeben

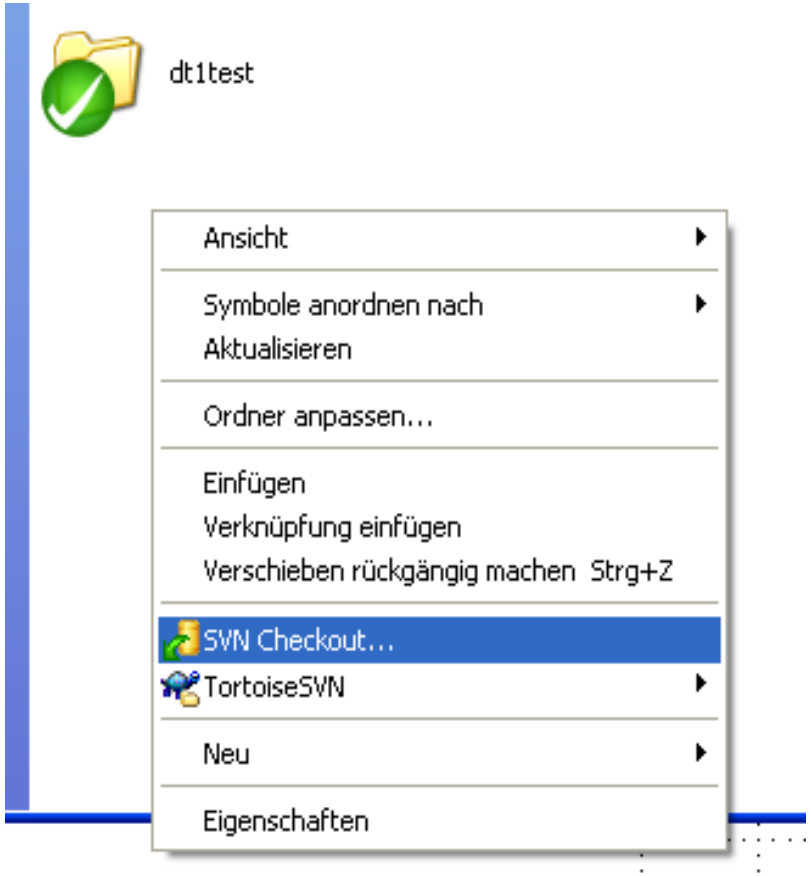


3. Zum Verzeichnis browsen und auschecken (menu rechte Maustaste)

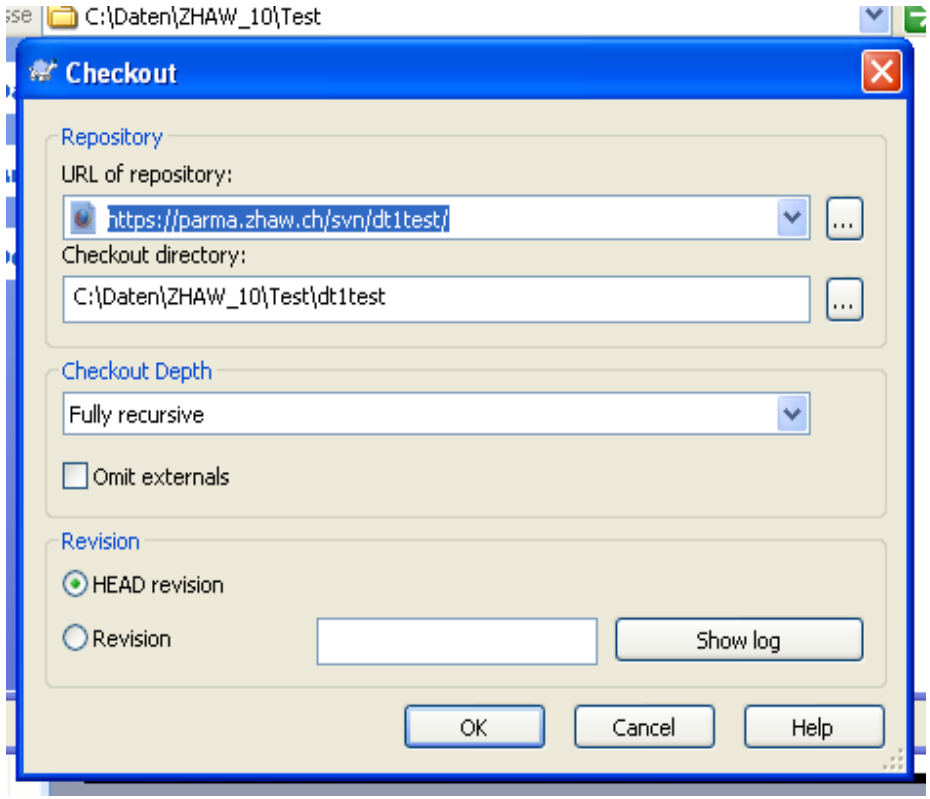


Projekt Daten Verwalten mit Tortoise:
Ein bestimmtes Verzeichnis aus einem Repository auschecken (2)

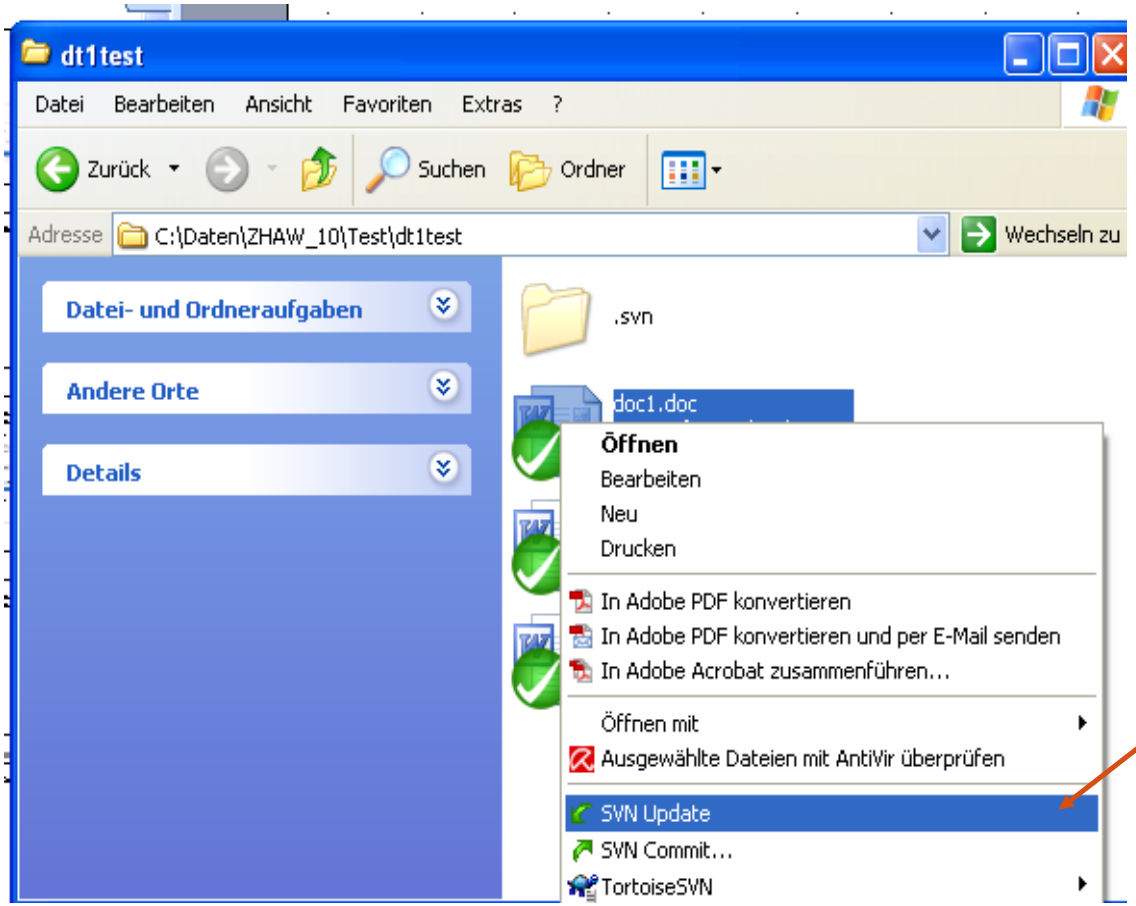
3. Check-Out aufrufen



4. URL des Verzeichnis eingeben

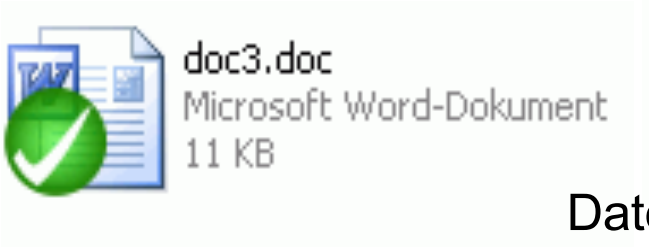


Projekt Daten Verwalten mit Tortoise:
Dateien aus dem Repository aktualisieren



SVN Update wählen

Projekt Daten Verwalten mit Tortoise: **Symbole**



Datei aktuell

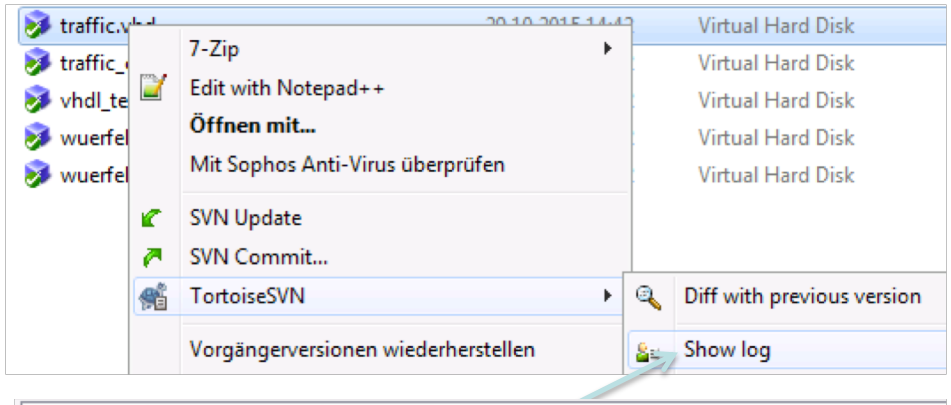
Datei wurde vor Ort verändert und muss eingecheckt werden, damit andere die Datei auschecken können

Datei wurde vor Ort neu angelegt und ist noch nicht im Repository enthalten

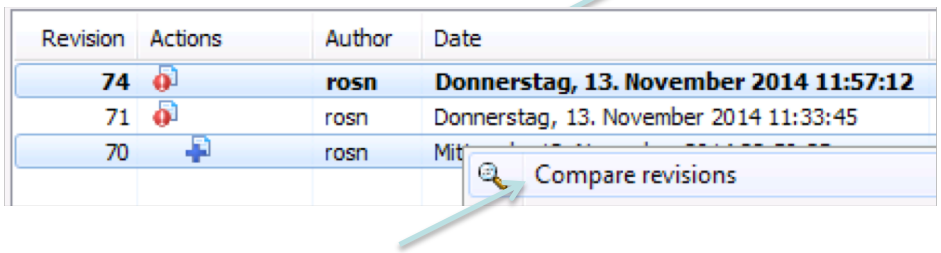
Konflikt: Eine oder mehrere lokale Dateien und Dateien im Repository haben unterschiedlichen Inhalt. Die Dateien erhalten beim auschecken unterschiedliche Namen, damit sie verglichen werden können.

Alte Versionen eines Files

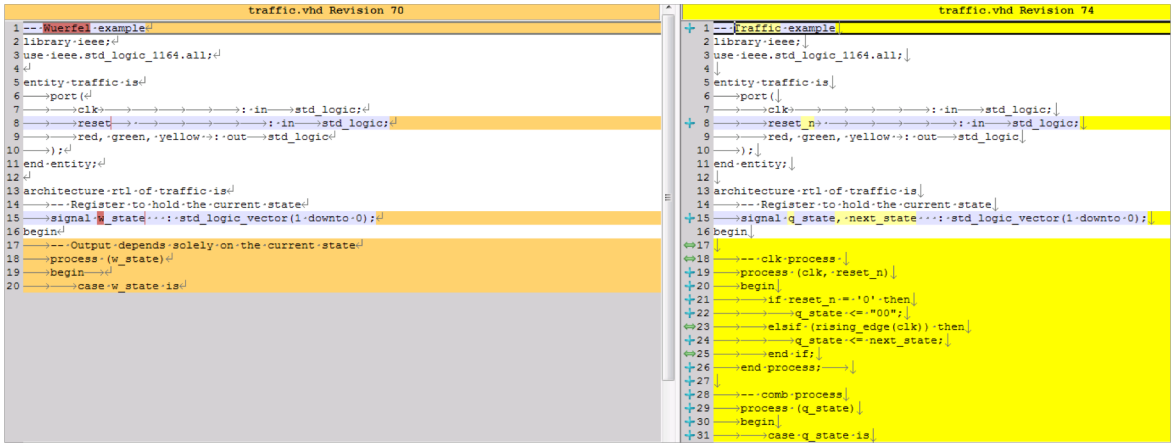
- Show Log aufrufen



- Verschiedene Versionen mit “Ctrl” selektieren und “Compare revisions” aufrufen



- Files vergleichen: “Compare Files”



Projekt Daten Verwalten mit Tortoise:
Was noch beachtet werden muss

- Dateien löschen nur über Tortoise
- Dateien umbennen nur über Tortoise
- Wenn Gefahr besteht, dass mehrere Personen gleichzeitig die gleichen Dateien bearbeiten, mit „Lock“ auschecken