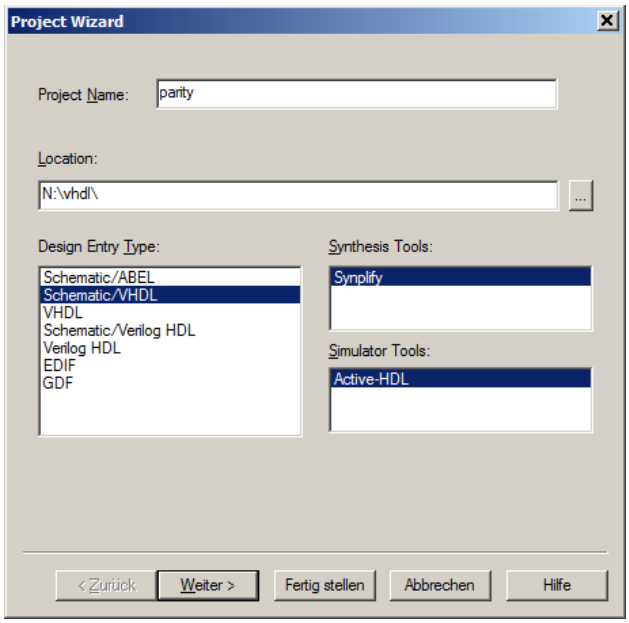
VHDL-Projekt

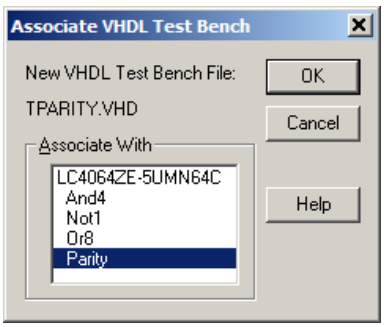
Dieses kleine Laborprojekt vermittelt Ihnen einen ersten Eindruck, wie sich mit Hilfe der Sprache VHDL Hardware-Schaltungen entwickeln lassen. In den folgenden Aufgaben werden Sie zunächst eine Beispielschaltung mit einem VHDL-Simulator simulieren und anschließend weiterentwickeln.

Aufgabe 1: Anlegen eines VHDL-Projekts

Erzeugen Sie zunächst ein Projektverzeichnis auf dem **Netzlaufwerk N:** mit dem Namen **VHDL**. Legen Sie dort zwei VHDL-Dateien mit den im Anhang abgedruckten Inhalten an. Die erste Datei enthält die Beschreibung der Hardware-Schaltung und die zweite die Testbench. Um sich Tipparbeit zu ersparen, können Sie die Dateien von der Webseite der TI-1-Vorlesung herunterladen. Starten Sie anschließend das Werkzeug **ispLEVER Classic Project Navigator** Version 1.6 und erstellen Sie mit dem Projekt-Wizard ein VHDL-Projekt (Menüleiste: File /New Projekt):



Klicken Sie auf die Schaltfläche „Fertig stellen“.

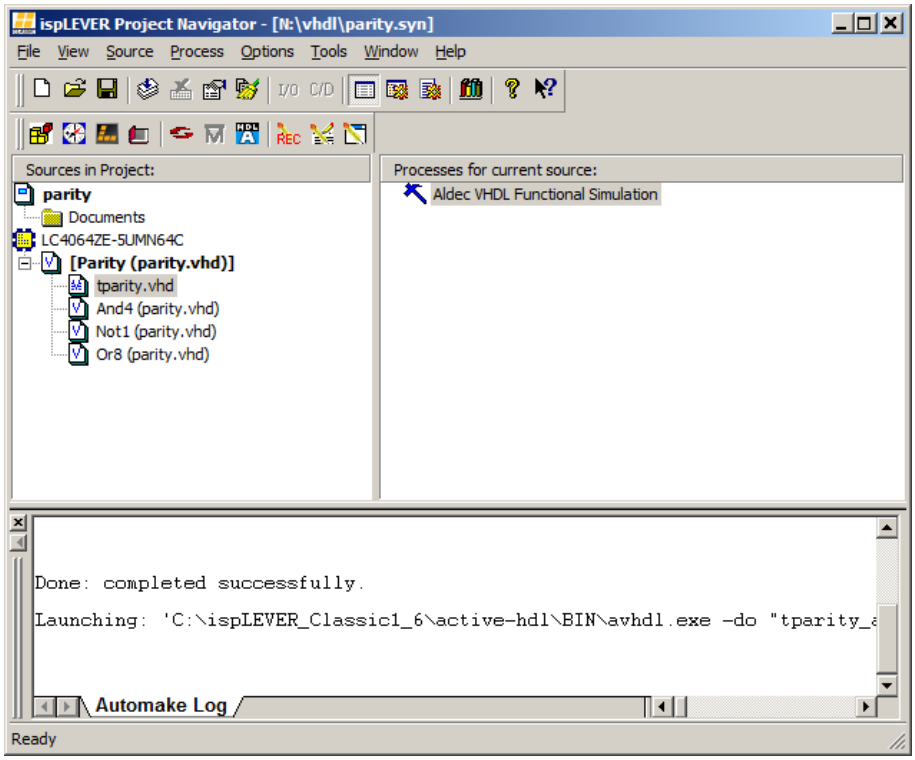


Fügen Sie jetzt über den Menüpunkt „Source -> Import“ zuerst die Datei parity.vhd als VHDL-Modul und anschließend die Datei tparity.vhd als VHDL-TestBench zu Ihrem Projekt hinzu.

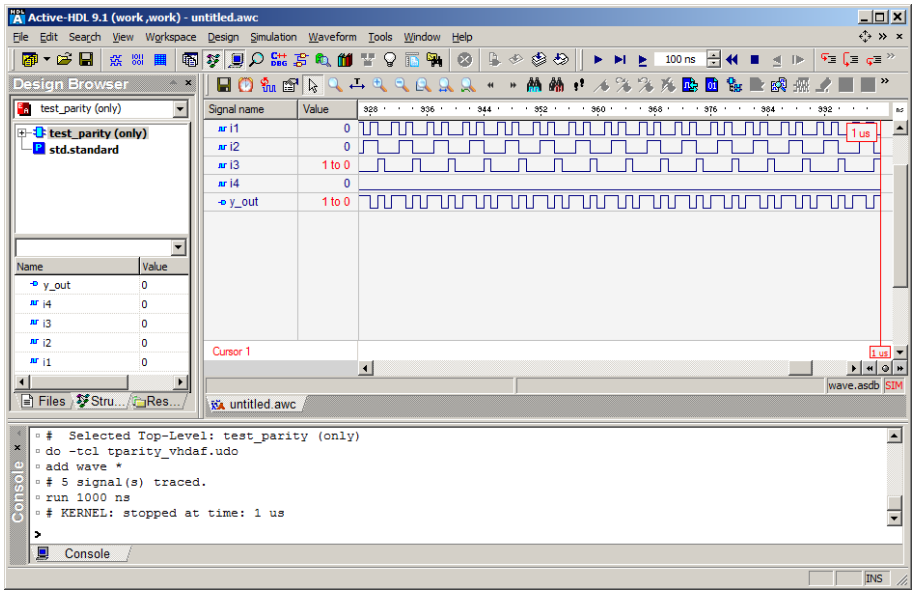
Beim Importieren der Testbench müssen Sie diese mit einer Komponente der Hardware-Schaltung assoziieren. Wählen Sie hier die (Haupt)komponente „parity“ aus.

Aufgabe 2: Compilieren und Simulieren

Wählen Sie Datei tparity.vhd aus. Über die Option „Aldec VHDL Functional Simulation“ können Sie die Hardware-Schaltung compilieren und den ActiveHDL-Simulator starten.



Über das erzeugte Zeitdiagramm (Waveform) können Sie jetzt die Schaltung analysieren:



Aufgabe 3: Vervollständigen der Testbench

Wie Sie wahrscheinlich bemerkt haben, ist die Testbench nicht vollständig (nicht alle Eingangskombinationen werden getestet). Modifizieren Sie die Testbench so, dass alle Eingabekombinationen geprüft werden. Simulieren Sie das Design erneut und kontrollieren Sie die Funktionsweise.

Aufgabe 4: Austausch der „Architecture“

In der Simulation wurde für die Entity „Parity“ die Architecture „Behavioral“ verwendet. Ein Blick auf die VHDL-Quellen zeigt, dass mit „Structural“ eine zweite Architecture vorhanden ist, die Sie anstelle der Architecture „Behavioral“ einbauen können. Ändern Sie die VHDL-Quellen so ab, dass ab jetzt die Architecture „Structural“ verwendet wird.

Aufgabe 5: Finden Sie den Fehler!

In die Architecture „Structural“ wurde mit Absicht ein Fehler eingebaut. Analysieren Sie das Signaldiagramm und versuchen Sie, den Fehler anhand der Werteverläufe aufzuspüren. Korrigieren Sie anschließend den Fehler in der VHDL-Datei.

Aufgabe 6: Konjunktive statt disjunktive Form

Die Architecture „Structural“ ist eine Eins-zu-Eins-Umsetzung der disjunktiven Normalform in eine Hardware-Schaltung. Berechnen Sie die konjunktive Normalform und erzeugen Sie anschließend eine dritte Architecture mit dem Namen „KNF“, die eine Eins-zu-Eins-Umsetzung der konjunktiven Normalform in Hardware darstellt. Testen Sie die Korrektheit Ihrer Schaltung, indem Sie den Parity-Generator mit der neuen Architecture simulieren.

Anhang

Datei „parity.vhdl“

```
-- Parity-Generator
--
-- 2007 Dirk W. Hoffmann
-- Hochschule Karlsruhe
-- ------------------------------------
entity Not1 is
  port(X : in bit;
       Y : out bit);
end;

architecture Only of Not1 is
begin
  Y <= not X;
end;
-- ------------------------------------
entity And4 is
  port(X4, X3, X2, X1 : in  bit;
                    Y : out bit);
end;

architecture Only of And4 is
begin
  Y <= X4 and X3 and X2 and X1;
end;
-- ------------------------------------
entity Or8 is
  port(X8, X7, X6, X5, X4, X3, X2, X1 : in  bit;
                                     Y : out bit);
end;

architecture Only of Or8 is
begin
  Y <= X8 or X7 or X6 or X5 or X4 or X3 or X2 or X1;
end;
-- ------------------------------------
entity Parity is
  port(X4,X3,X2,X1 : in  bit;
                 Y : out bit);
end;

architecture Behavioral of Parity is
begin
  with bit_vector'(X4,X3,X2,X1)
    select
      Y <= '0' when "0000",
           '1' when "0001",
           '1' when "0010",
           '0' when "0011",
           '1' when "0100",
           '0' when "0101",
           '0' when "0110",
           '1' when "0111",
           '1' when "1000",
```

```
           '0' when "1001",
           '0' when "1010",
           '1' when "1011",
           '0' when "1100",
           '1' when "1101",
           '1' when "1110",
           '0' when "1111";
end Behavioral;
-- -------------------------------------
architecture Structural of Parity is
  signal N1, N2, N3, N4                 : bit;
  signal S1, S2, S3, S4, S5, S6, S7, S8 : bit;

  component Not1
    port(X : in  bit;
         Y : out bit);
  end component;

  component and4
    port(X4, X3, X2, X1 : in  bit;
                      Y : out bit);
   end component;

   component or8
      port(X8,X7,X6,X5,X4,X3,X2,X1 : in  bit;
                                  Y : out bit);
   end component;

begin
  Inv1 : Not1 port map (X1,N1);
  Inv2 : Not1 port map (X2,N2);
  Inv3 : Not1 port map (X3,N3);
  Inv4 : Not1 port map (X4,N4);

  Minterm1 : And4 port map (N4,X3,X2,X1,S1);
  Minterm2 : And4 port map (X4,N3,X2,X1,S2);
  Minterm3 : And4 port map (X4,X3,N2,X1,S3);
  Minterm4 : And4 port map (X4,X3,N2,N1,S4);
  Minterm5 : And4 port map (N4,N3,N2,X1,S5);
  Minterm6 : And4 port map (N4,N3,X2,N1,S6);
  Minterm7 : And4 port map (N4,X3,N2,N1,S7);
  Minterm8 : And4 port map (X4,N3,N2,N1,S8);

  Final : or8 port map(S8,S7,S6,S5,S4,S3,S2,S1,Y);

end Structural;
```

Datei „tparity.vhdl“

```
-- 2007 Dirk W. Hoffmann
-- Hochschule Karlsruhe
--
-- Testbench for the "parity" design
-- ------------------------------------
entity test_parity is
    PORT ( y_out : OUT bit);
end;

architecture only of test_parity is
SIGNAL i4 : bit := '0';
SIGNAL i3 : bit := '0';
SIGNAL i2 : bit := '0';
SIGNAL i1 : bit := '0';

COMPONENT parity
      port(in_4,in_3,in_2,in_1 : in  bit;
                               y : out bit);
END COMPONENT ;

for dut : parity use entity work.parity(Behavioral)
    port map (x4=>in_4, x3=>in_3, x2=>in_2, x1=>in_1, y=>y);

begin
dut : parity
   port map (
   in_4 => i4,
   in_3 => i3,
   in_2 => i2,
   in_1 => i1,
   y => y_out);

stimulus : process
  begin
    i4<='0'; i3<='0'; i2<='0'; i1<='0';
    wait for 1 ns;
    i4<='0'; i3<='0'; i2<='0'; i1<='1';
    wait for 1 ns;
    i4<='0'; i3<='0'; i2<='1'; i1<='0';
    wait for 1 ns;
    i4<='0'; i3<='0'; i2<='1'; i1<='1';
    wait for 1 ns;
    i4<='0'; i3<='1'; i2<='0'; i1<='0';
    wait for 1 ns;

end process stimulus;

end only;

--end
```