VHDL Übersicht

Die wichtigsten VHDL Syntax-Elemente auf einen Blick

Entity:

```
library ieee;
use ieee.logic_1164.all;
use ieee.numeric_std.all;

entity MyModule is
  generic (
      S : std_logic_vector (3 downto 0) := "0101";
      N : integer := 8
      );
  port (
      clk    : in  std_logic;
      reset  : in  std_logic;
      dataIn : in  std_logic_vector (3 downto 0);
      result : out std_logic_vector (N-1 downto 0)
      );
end;
```

Architecture:

```
architecture rtl of MyModule is
      -- Signal- und Typdefinitionen vor "begin" !
      constant ci  : integer := 7;
      signal   tmp : unsigned (4 downto 0);

      type T_STATE is (Init,Count,Open,Alarm);
      signal Zustand :  T_STATE;
begin
      -- Prozesse
      -- Signal- bzw. Portzuweisungen
      ...
end;
```

Prozesse:

Asynchroner Prozess

```
process (sig1,dataIn) -- Sensitivitätsliste!
      variable my_var : std_logic_vector (3 downto 0);
begin
      -- Anweisungen, z.B. if, case
end process;
```

Synchroner Prozess *(mit synchronem Reset)*

```
process -- Keine Sensitivitätsliste!
      variable my_other_var : unsigned (3 downto 0);
begin
      wait until rising_edge (clk);
      if reset = '1' then
            -- Resetverhalten
      else
            -- Anweisungen, z.B. if, case
      end if;
end process;
```

Synchroner Prozess *(mit asynchronem Reset)*

```
process (clk, reset) -- Sensitivitätsliste!
begin
      if reset = '1' then
            -- Resetverhalten
      elsif rising_edge (clk)
            -- Anweisungen, z.B. if, case
      end if;
end process;
```

Generell: *Entweder Sensitivitätsliste oder Wait-Anweisung*
Für Synthese: max. 1 Wait-Anweisung je Prozess

Instanziierung:

```
my_module_instance: entity work.MyModule
  generic map (
      S => "1101",
      N => 4
      )
  port map (
      clk    => clk_sig,  -- die Port-Namen aus
      reset  => reset,    -- der Entity stehen links
      dataIn => data_inp,
      result => res_out
      );
```

Bedingte Ausführung *(nur in Prozessen)*:

```
if Bedingung then
      ...
elsif Bedingung then
      ...
else
      ...
end if;

case Ausdruck is
   when "000" =>
      ...
   when "010" | "111" =>
      ...
   when others =>
      ...
end case;
```

Schleifen *(nur in Prozessen)*:

```
for i in 0 to 7 loop
      ...
end loop;

while Bedingung loop
      ...
end loop;
```

Datentypen:

integer	32 bit, vorzeichenbehaftet
std_logic std_logic_vector **unsigned, signed**	: 9-wertige Datentypen (u.a.: **0,1,U,X,-**)

Typumwandlung:

Integer nach (un)signed:	**to_(un)signed**(val,width)
(Un)signed nach Integer:	**to_integer**(val)
Std_logic_vector nach (un)signed:	(**un**)**signed**(val)
(Un)signed nach std_logic_vector:	**std_logic_vector**(val)

Signale vs. Variablen:

Variablen:	Zuweisung erfolgt sofort (Verhalten wie z.B. in C/C++)
Signale/Out-Ports:	Zuweisung erfolgt mit Unterbrechung/Ende des Prozesses *(nur die jeweils zuletzt ausgeführte Zuweisung wird "sichtbar")*
Syntax der Zuweisung:	`my_var := "1010";  my_sig <= "1010";` (*Variablen* `:=` *Signale, Ports* `<=`)

Operatoren:

Symbol	Bedeutung	Datentypen	Rangfolge
abs	Absolutwert	integer, (un)signed	1 (hoch)
not	Invertierung	(un)signed, std_logic(_vector)	
**	Potenzierung	integer	
* /	Multiplikation, Division	integer, (un)signed	2
mod,rem	Modulo , Rest	integer, (un)signed	
+ -	Vorzeichen	integer, (un)signed	3
+ -	Addition, Subtraktion	integer, (un)signed	4
&	Vektoren zusammenfügen	(un)signed, std_logic(_vector)	
sll,sla,sra, srl,rol,ror	Schieben, Rotieren	(un)signed, std_logic_vector	5
= /=	Gleich, Ungleich	integer, (un)signed, std_logic(_vector)	6
< <= > >=	Kleiner/Größer	integer, (un)signed	
and or nand nor xor	Logische Verknüpfungen	(un)signed, std_logic(_vector)	7 (niedrig)

Konstanten *(Beispiel: Zahlenwert 254):*

Integer: 254 10#254# 16#FE#
Vektor: "11111110" x"FE"