Beginn 10:00

Foliensatz zum Download

http://dodo.fb06.fh-muenchen.de/hermann/digital/mfm_digital_v5.pdf

Details zu VHDL in den beiden Skripten

Tipp

Folie 9 (Beispiel Zählerkette) ausdrucken, wird in der VL benutzt, aber dann durch Diamond meist verdeckt!

VHDL – Numerik

Datentypen für Zahlen

- Fließkomma, nicht für den Schaltungsentwurf: real
- Ganze Zahlen

integer		int32_t
natural	$x \geq 0$	uint32_t
positive	$x \geq 1$	uint32_t

Deklaration mit optionaler Bereichsangabe

```
signal x1: integer;
signal x2: integer  range -100 to 100;
signal x3: positive range   50 to   99;
```

Operationen und Konstanten

Operationen

+, -, <, <=, =, >=, >, /=	problemlos
* (Multiplikation)	hoher Aufwand
/ (Division)	i.d.R. nicht für den Entwurf

Konstanten

17	Basis 10
2#101#	Basis 2, Dualzahl 101 (in C: 0b101)
16#d5a#	Basis 16, Hexadezimalzahl d5a (in C: 0xd5a)

Vektoren als Zahlen

- **Mit Vektoren stehen Bitfelder beliebiger Länge zur Verfügung**
  - Die Richtung downto passt zur üblichen Stellenwertigkeit
  - aber: bit_vector und std_logic_vector haben keinen Zahlenwert

- **Standard-Package numeric_bit bzw. numeric_std**
  - Definition neuer Vektoren unsigned und signed
  - Benutzung und Syntax wie bit_vector und std_logic_vector
  - Interpretation als vorzeichenlose oder vorzeichenbehaftete Zahl möglich
  - Operationen wie auf integer erlaubt

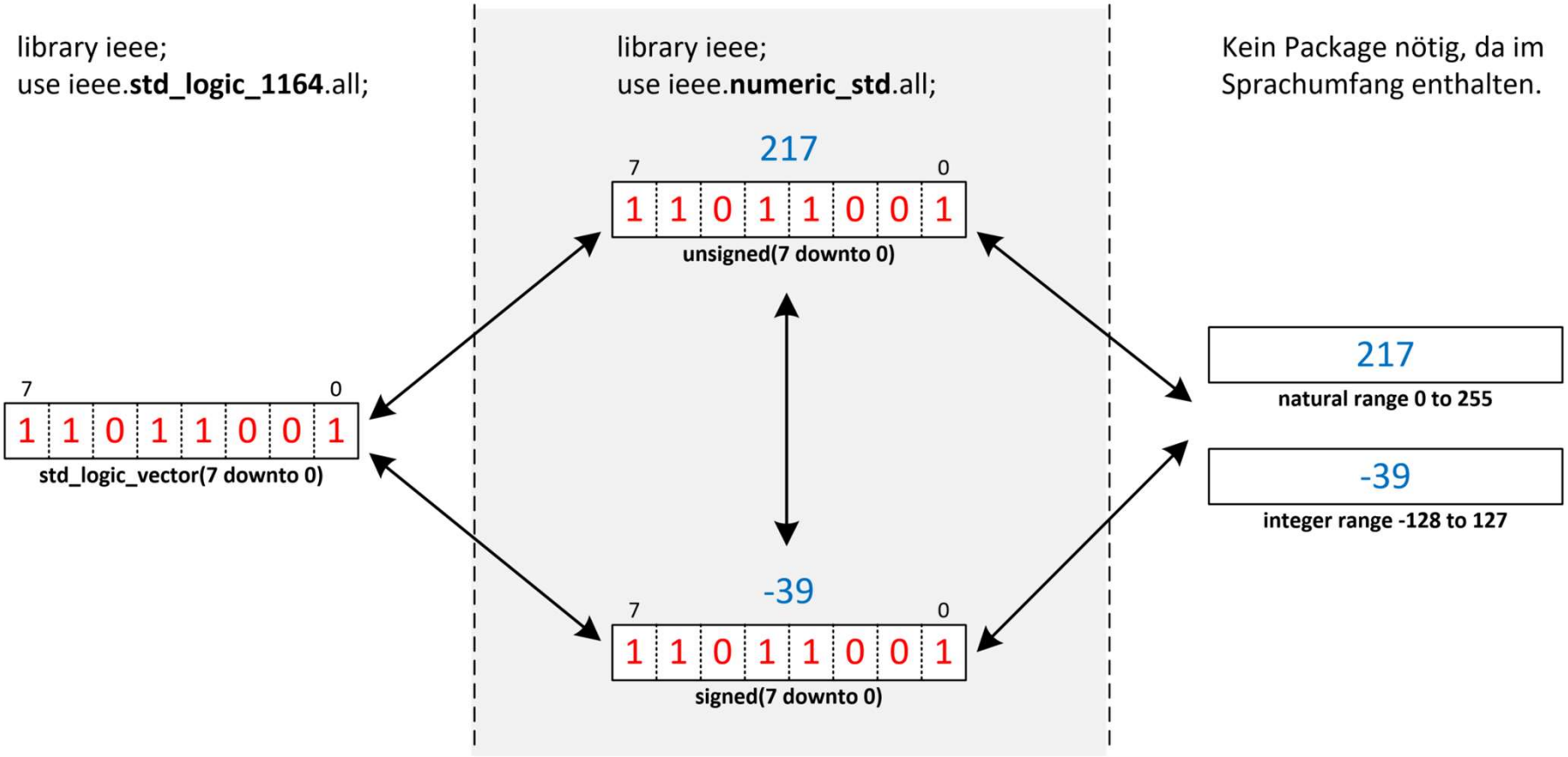
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

signal x: unsigned(6 downto 0);  -- Einzelelemente std_logic, Zahlenbereich [0, 127]

x <= x + 1;                      -- Addition einer 1 auf den Vektor x
```

Vergleich Zahl - Vektor

- **Zahl:**
  - integer/natural/positive haben nur Zahlenwerte
  - Der Range gibt dem Compiler an, wie viele Stellen (bits) benötigt werden, braucht der Anwender nicht zu sagen
  - Über-/Unterlauf des Range wird nur bei der Simulation geprüft, nicht in der Schaltung

- **Vektor**
  - Der Range (a downto b) definiert den Zahlenbereich
  - Zugriff auf einzelne Stellen (Bits) ist möglich

Konversionen Vektor/Zahl



Konversionen Vektor/Zahl

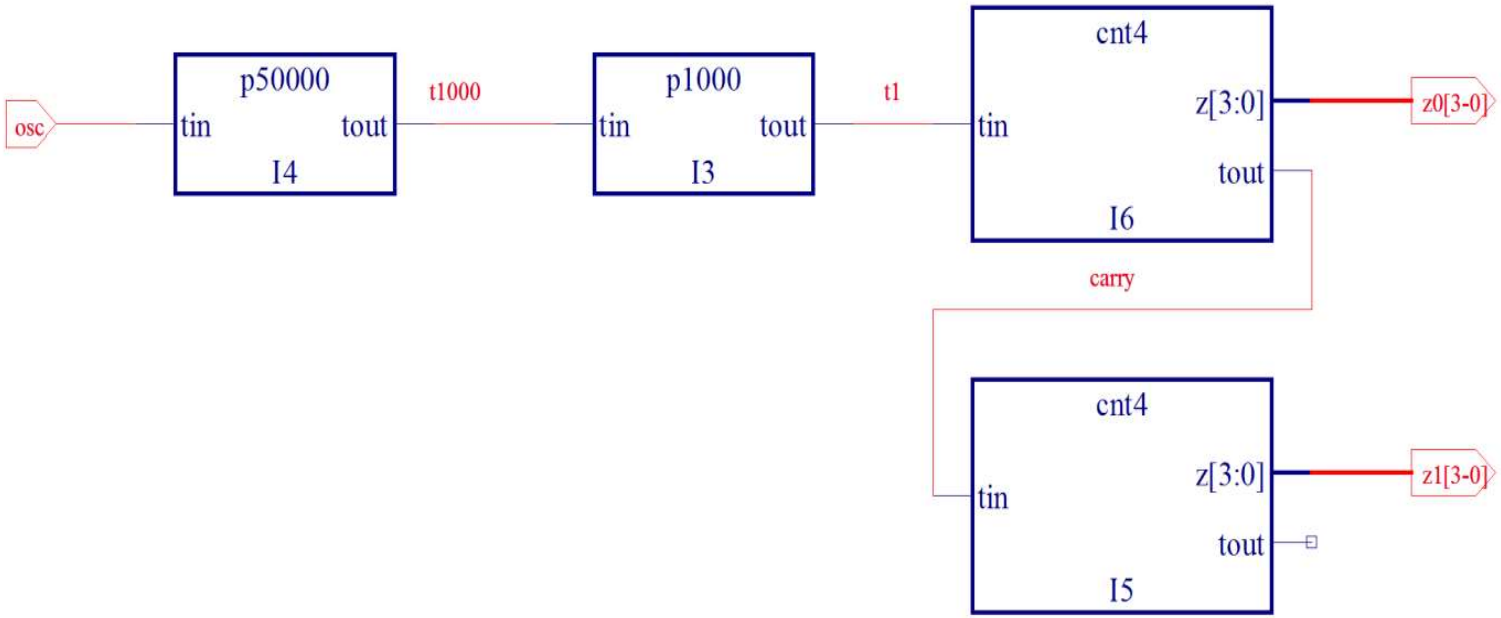
von	nach	Funktion
std_logic_vector	unsigned	unsigned()
std_logic_vector	signed	signed()
unsigned	std_logic_vector	std_logic_vector()
unsigned	signed	signed()
unsigned	integer, natural, positive	to_integer()
signed	std_logic_vector	std_logic_vector()
signed	unsigned	unsigned()
signed	integer	to_integer()
integer, natural, positive	unsigned	to_unsigned(zahl,bits)
integer, natural, positive	signed	to_signed(zahl,bits)

```
signal v: unsigned(6 downto 0);            -- Vektor v
signal z: natural range 0 to 100;          -- Zahl   z

z             <= to_integer(v);
v(1 downto 0) <= to_unsigned(z, 2);        -- Nur die Bits 1 und 0 von z werden übertragen
```

Strukturbeschreibung

- **Die Strukturbeschreibung entspricht einem Schaltplan bzw. Blockdiagramm**
  - In einer architecture können in der nebenläufigen Umgebung beliebig viele Blöcke enthalten sein
  - Blöcke werden durch lokale Signale verbunden
  - Blöcke können parametrisiert werden (Compilezeit)
  - Blöcke können als Komponenten (Platzhalter) deklariert werden (-> Black Box möglich)
  - Hierarchie beliebig tief
  - Die Zuordnung eines Block zu einer entity ist sehr flexibel (Konfigurationen)

Beispiel: Zählerkette



Package

- **Entspricht einem Header in C**
- **Deklarationsteil (immer vorhanden)**
  - package name is / end name;
  - Deklaration von Datentypen, Konstanten, Funktionen, Komponenten
- **Definitionsteil (nur falls nötig)**
  - package body name is / end name;
  - Definitionen von Funktionen
- **Einbindung in andere Design Units**
  - library work; -- wo zu finden, hier in work (Projekt)
  - use work.package.all; -- was wird eingebunden (hier alles)