ERA-Tutorium 11

Thomas Kilian

Frage zur Vorlesung

Frage zur Vorlesung

- Was ist der Unterschied zwischen Schaltnetzen und -werken?

Frage zur Vorlesung

- Was ist der Unterschied zwischen Schaltnetzen und -werken?

  ➡ Schaltnetze speichern keine Informationen

VHDL

VHDL

- **V**ery **H**igh Speed Integrated Circuit Hardware **D**escription **L**anguage

VHDL

- **V**ery **H**igh Speed Integrated Circuit Hardware **D**escription **L**anguage

- beschreibt Struktur und Verhalten von Schaltungen

VHDL

- **V**ery **H**igh Speed Integrated Circuit Hardware **D**escription **L**anguage
- beschreibt Struktur und Verhalten von Schaltungen
- Beschreibungssprache

Grundlagen: VHDL

Grundlagen: VHDL

- Signale: signal <name> : <type> [:=default];

Grundlagen: VHDL

- Signale: signal <name> : <type> [:=default];
  - In etwa ein Kabel (bzw. Ausgang/Eingang)

Grundlagen: VHDL

- Signale: signal <name> : <type> [:=default];
  - In etwa ein Kabel (bzw. Ausgang/Eingang)
- Types:

Grundlagen: VHDL

- Signale: signal <name> : <type> [:=default];
  - In etwa ein Kabel (bzw. Ausgang/Eingang)
- Types:
  - std_logic: U, X, 0, 1, Z, W, L, H

Grundlagen: VHDL

- Signale: signal <name> : <type> [:=default];
  - In etwa ein Kabel (bzw. Ausgang/Eingang)
- Types:
  - std_logic: U, X, 0, 1, Z, W, L, H
  - std_logic_vector: mehrere einzelne Signale

Grundlagen: VHDL

- Signale: signal <name> : <type> [:=default];
  - In etwa ein Kabel (bzw. Ausgang/Eingang)
- Types:
  - std_logic: U, X, 0, 1, Z, W, L, H
  - std_logic_vector: mehrere einzelne Signale
  - std_logic_[un]signed: auch Vektoren

Grundlagen: VHDL

Grundlagen: VHDL

- Deklarationen

Grundlagen: VHDL

- Deklarationen
  - signal sig: std_logic := ‘1’;

Grundlagen: VHDL

- Deklarationen
  - signal sig: std_logic := ‘1’;
  - signal vec: std_logic_vector (5 downto 0);

Grundlagen: VHDL

- Deklarationen
  - signal sig: std_logic := ‘1’;
  - signal vec: std_logic_vector (5 downto 0);
  - signal vec: std_logic_vector (0 to 5);

Grundlagen: VHDL

- Deklarationen
  - signal sig: std_logic := ‘1’;
  - signal vec: std_logic_vector (5 downto 0);
    - signal vec: std_logic_vector (0 to 5);
- Zugriff

Grundlagen: VHDL

- Deklarationen
  - signal sig: std_logic := ‘1’;
  - signal vec: std_logic_vector (5 downto 0);
    - signal vec: std_logic_vector (0 to 5);
- Zugriff
  - vec(3) := 4. Signal von rechts

Grundlagen: VHDL

- Deklarationen
  - signal sig: std_logic := ‘1’;
  - signal vec: std_logic_vector (5 downto 0);
  - signal vec: std_logic_vector (0 to 5);
- Zugriff
  - vec(3) := 4. Signal von rechts
  - ver(2 downto 1) := 2 Signale

Grundlagen: VHDL

Grundlagen: VHDL

- Zuweisungen

Grundlagen: VHDL

- Zuweisungen
  - a <= e;

Grundlagen: VHDL

- Zuweisungen
  - a <= e;
  - a <= e when <bedingung1> ‘1’ when <bedingung2> else ‘0’;

Grundlagen: VHDL

- Zuweisungen
  - a <= e;
  - a <= e when <bedingung1> ‘1’ when <bedingung2> else ‘0’;
  - Mögliche Bedingungen: siehe Merkblatt

Grundlagen: VHDL

- Zuweisungen
  - a <= e;
  - a <= e when <bedingung1> ‘1’ when <bedingung2> else ‘0’;
  - Mögliche Bedingungen: siehe Merkblatt
- Beispiele:

Grundlagen: VHDL

- Zuweisungen
  - a <= e;
  - a <= e when <bedingung1> ‘1’ when <bedingung2> else ‘0’;
  - Mögliche Bedingungen: siehe Merkblatt
- Beispiele:
  - a <= not e; (btw: a <= nOt E;)

Grundlagen: VHDL

- Zuweisungen
  - a <= e;
  - a <= e when <bedingung1> ‘1’ when <bedingung2> else ‘0’;
  - Mögliche Bedingungen: siehe Merkblatt
- Beispiele:
  - a <= not e; (btw: a <= nOt E;)
  - a <= b when **b xor “101”** else “000”;

concurrent vs. sequential statements

- **concurrent**: wird gleichzeitig ausgeführt

- **sequential**: execution in order of appearance, z.B. in Prozessen

VHDL: entity

Entity
entity *entityname* **is** **port (** *portname* **:** *modus datatype* **;** **...** *portname* **:** *modus datatype* **);** **end** *entityname* **;**

VHDL: architecture

Architecture

```
architecture versionname of entityname is
        signal signalname : datatype ;
        ...
begin
        {concurrent statements}
        {component instantiations}
        {processes}
end versionname;
```

VHDL: process

Process

`process(`*sensitivitylist*`)`
`begin`
`{`*statements*`}`
`end process;`

VHDL: other

if-statement

```
if condition then
        ...
else
        ...
end if;

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

signal-assignment statement

```
signalname <= term;
```

case-statement

```
case signalname is
when value => {statements}
   ...
when others => {statements}
end case;
```

VHDL: example

```
-- AND-Gate
library IEEE;
use IEEE.STD_LOGIC_1164.all;

-- define entity (i/o)
entity ANDGATE is
    port (
        E1, E2: in STD_LOGIC;
        A: out STD_LOGIC
    );
end entity;
-- alternativ: "end ANDGATE;"

architecture v1 of ANDGATE is
    begin
        A <= E1 and E2;
end architecture;
-- alternativ: "end v1;"
~
```

Aufgabe 1

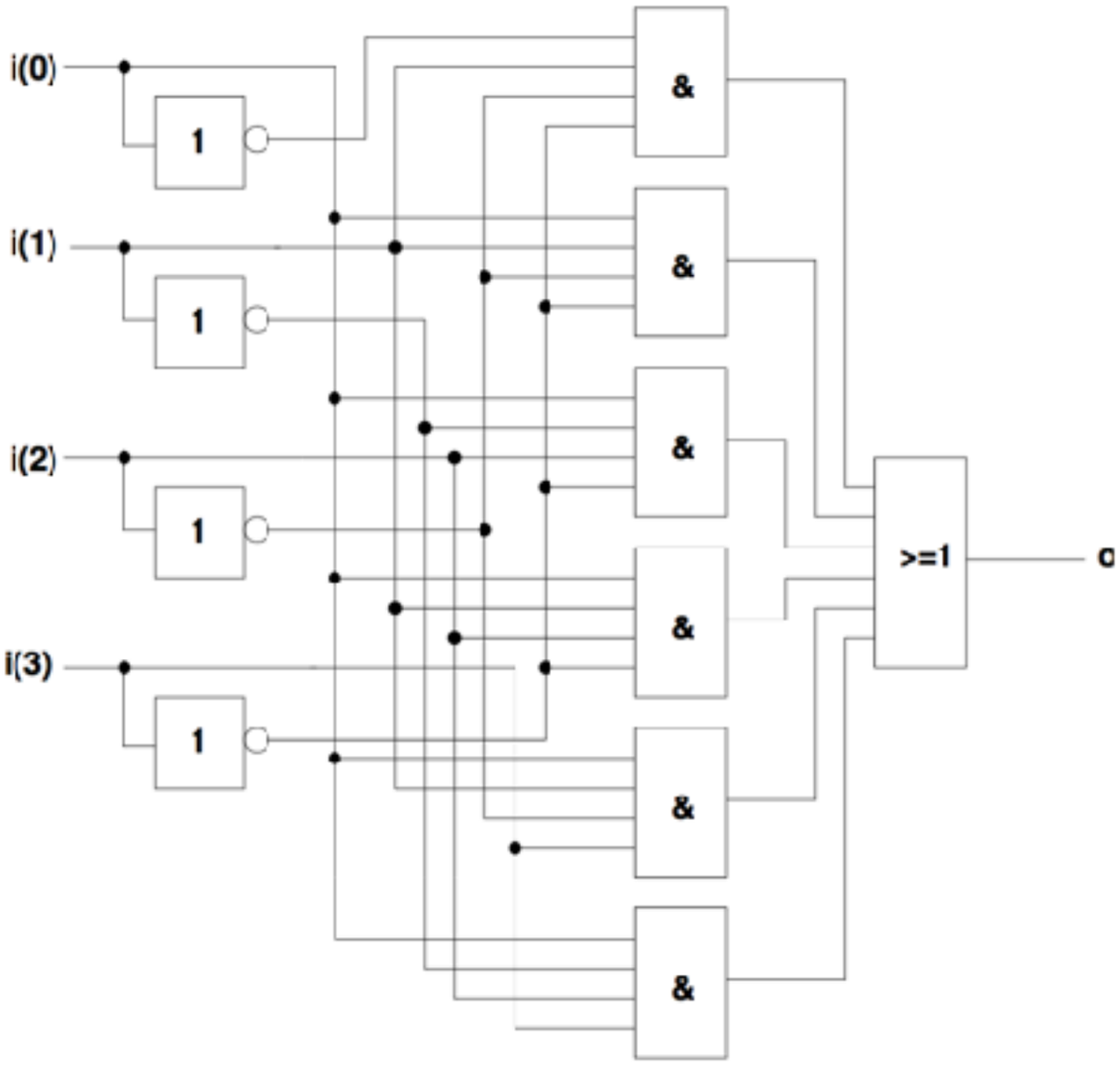
- signal a: std_logic_vector(7 downto 0);
- signal c: std_logic_vector(7 downto 0);

1. Multipliziere a mit 4, teile c durch 8!
2. Schaltplan für 1.

Aufgabe 2

- Was ist mächtiger:
  - Wahrheitstabelle mit 4 Eingängen
  - DNF mit 4 or-Termen und beliebig vielen and-Termen

Aufgabe 3



1. Ausdruck für Schaltung?

2. concurrent statement in VHDL mit signal i: std_logic_unsigned (3 downto 0);

3. Wahrheitstabelle

4. Was macht die Schaltung?

5. Anderer Ausdruck für die Schaltung in VHDL?

Aufgabe 4

Auflösungsfunktion von VHDL

Auszug aus std_logic_1164-body.v93 (github.com/tgingold/ghdl)

```
28      constant resolution : table_2d :=
29      --  UX01ZWLH-
30        ("UUUUUUUUU",   --  U
31         "UXXXXXXXX",   --  X
32         "UX0X0000X",   --  0
33         "UXX11111X",   --  1
34         "UX01ZWLHX",   --  Z
35         "UX01WWWWX",   --  W
36         "UX01LWLWX",   --  L
37         "UX01HWWHX",   --  H
38         "UXXXXXXXX"    --  -
39         );
```