Entwurf und Verifikation digitaler Systeme mit VHDL

Wolfgang Günther

Infineon AG

CL DAT DF LD V

guenther@informatik.uni-freiburg.de,

wolfgang.guenther@infineon.com

Inhalt der Vorlesung

- *Der Entwurfsprozess*
- *Die Sprache VHDL*
  - *Abstraktionsebenen*
  - *Datentypen*
  - *Befehle*
- *Modellierung von FSMs*
- *Verifikation digitaler Schaltungen*

Vorbemerkungen

- *Zielrichtung der Vorlesung:*
  - *Synthese*
  - *Verifikation*
  - *Beispiele*
- *Keine vollständige Beschreibung von VHDL*
  - *Konzentration auf "synthetisierbares" VHDL*

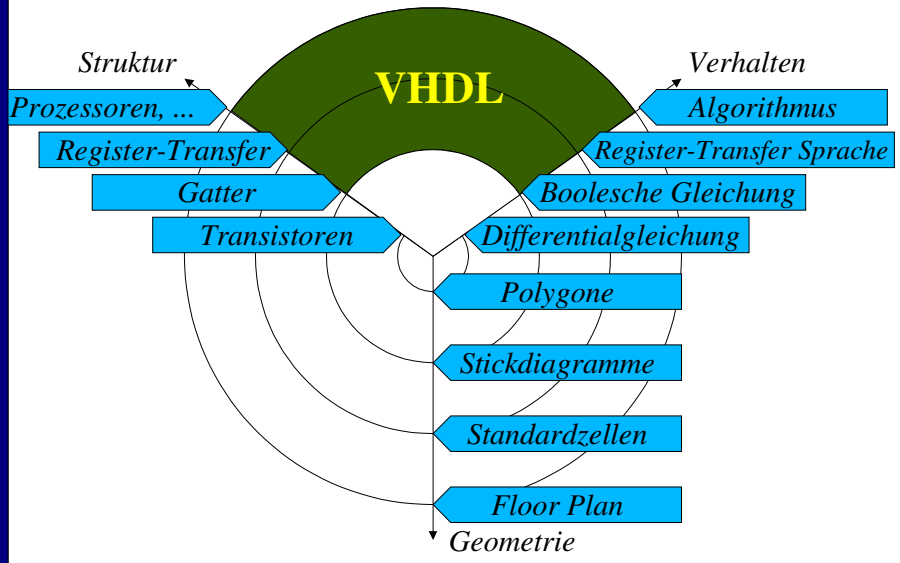
Literatur

- *P. J. Ashenden: The Designer's Guide to VHDL. Morgan Kaufmann Publishers, 1995.*
- *D. Perry: VHDL. McGraw-Hill, 1998.*
- *K. C. Chang: Digital Design and Modeling with VHDL and Synthesis. IEEE Computer Society Press, 1997.*
- *IEEE Standard VHDL Language Reference Manual (schwierig zu lesen)*

VHDL

- *Very High Speed Integrated Circuits Hardware Description Language*
- *Beschreibung komplexer elektronischer Systeme*
  - *Austauschbarkeit*
  - *Wiederverwendbarkeit*
- *Industriestandard*
- *Beschreibung auf verschiedenen Abstraktionsebenen*

Abstraktionsebenen



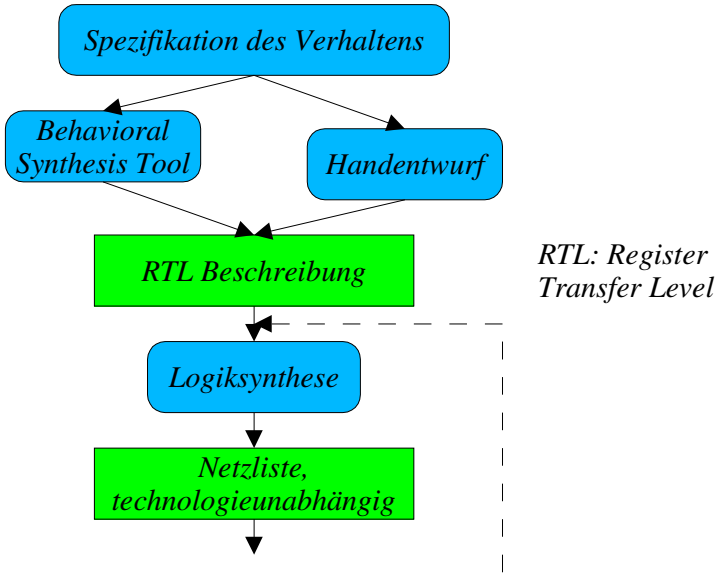
Wichtige Eigenschaften

- *Modellierung auf Verhalten- und auf Strukturebene*
- *Modellierung von Gleichzeitigkeit*
- *Abstrakte Datentypen*
- *Hochsprachenkonstrukte*
- *Strenge Typisierung*
- *Diskrete Event-Simulation*

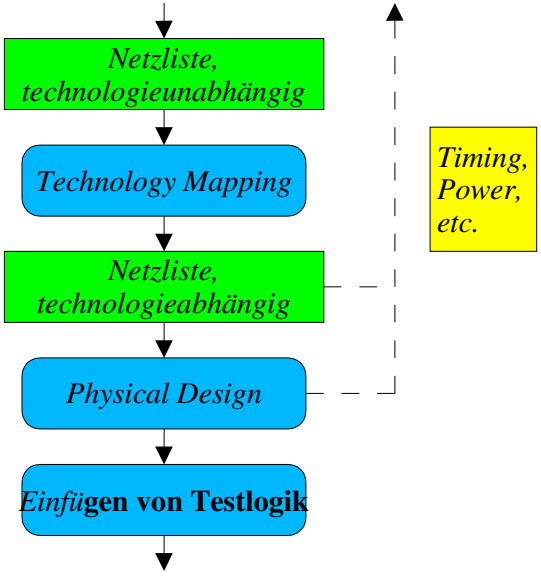
Standards

- *Standardisierung 1987 als IEEE 1076*
- *Erweitert 1993 (IEEE 1076-1993)*
- *IEEE 1076.1: Analog Extensions to VHDL*
- *IEEE 1076.2: Mathematical Package*
- *IEEE 1076.3: Synthesis Package*
- *IEEE 1076.4: Timing Methodology (VITAL)*
- *IEEE 1076.5: VHDL Utility Library*

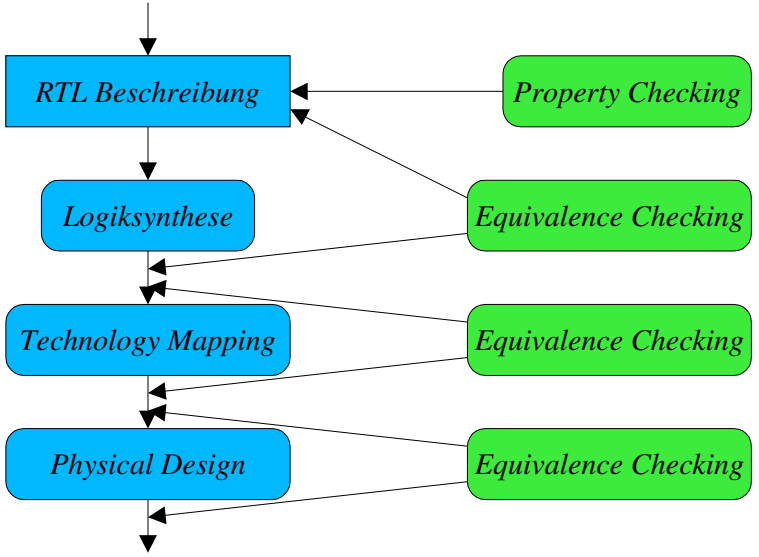
Der Entwurfsprozess



Der Entwurfsprozess (2)



Formale Verifikation



Bezeichner (identifier)

- *Dürfen nur aus Buchstaben, Ziffern und Unterstrichen bestehen*
- *müssen mit einem Buchstaben beginnen*
- *jedem Unterstrich muss ein anderes Zeichen folgen*
- *es dürfen keine reservierten Worte verwendet werden*
- *Groß-Kleinschreibung spielt keine Rolle*

Beispiele

- *Erlaubt:*
  - `bus, Bus, BUS`
  - `bus_7`
  - `mem_7_reg`
- *Nicht erlaubt:*
  - `bus__7`
  - `7mem`
  - `_mem`

Reservierte Worte VHDL93

group	*ror*
impure	*shares*
inertial	*sla*
literal	*sll*
postponed	*sra*
pure	*srl*
reject	*unaffected*
rol	*xnor*

Reservierte Worte VHDL

abs	*component*	*if*	*of*	*select*	*xor*
access	*configuration*	*in*	*on*	*severity*	
after	*constant*	*inout*	*open*	*signal*	
alias	*disconnect*	*is*	*or*	*subtype*	
all	*downto*	*label*	*others*	*then*	
and	*else*	*library*	*out*	*to*	
architecture	*elsif*	*linkage*	*package*	*transport*	
array	*end*	*loop*	*port*	*type*	
assert	*entity*	*map*	*procedure*	*units*	
attribute	*exit*	*mod*	*process*	*until*	
begin	*file*	*nand*	*range*	*use*	
block	*for*	*new*	*record*	*variable*	
body	*function*	*next*	*register*	*wait*	
buffer	*generate*	*nor*	*rem*	*when*	
bus	*generic*	*not*	*report*	*while*	
case	*guarded*	*null*	*return*	*with*	

Extended Identifiers

- *Beginnen und enden mit Backslash (\')*
- *Dürfen dazwischen beliebige Zeichen enthalten*
- *Sind case-sensitiv*
- *Beispiele:*

```
\data bus\    \clock #7\     \start__\

name     \name\     \Name\     \name \
```

Kommentare

- *Beginnen mit „--" und enden mit der Zeile*
- *Beispiel:*

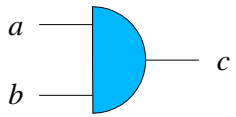
```
s <= '0'; -- Kommentar
```

- **Bemerkung**: *Kommentare können Anweisungen für die Synthese enthalten, etwa*

```
-- pragma synthesis_off
-- pragma synthesis_on
```


Beispiel: AND-Gatter

- *Strukturell:*



- *Wertetabelle:*

b\a	0	1
0	0	0
1	0	1

Entity für ein AND-Gatter

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

entity my_and is
   port(a, b : in  std_logic;
        c    : out std_logic);
end my_and;
```


Entity

- *Definiert die Schnittstelle nach außen*
- *Pins können sein*
  - *in -- Eingang*
  - *out -- Ausgang*
  - *inout -- Bidirektional*
  - *... und andere*
- *Ausgänge können nicht gelesen werden, Eingänge können nicht geschrieben werden*

Der Datentyp std_logic

- *Definiert in ieee.std_logic_1164;*
- *Definition:*

```
type std_ulogic is ('U',  -- Uninitialized
                    'X',  -- Forcing unknown
                    '0',  -- Forcing zero
                    '1',  -- Forcing one
                    'Z',  -- High impedance
                    'W',  -- Weak unknown
                    'L',  -- Weak zero
                    'H',  -- Weak high
                    '-'); -- Don't care
```

- *Unterschied std_logic vs. std_ulogic: später*

Architecture

- *beschreibt die Funktion des Bausteins mit Hilfe von*
  - *Strukturelementen*
  - *Verhaltensbeschreibung*
  - *Mischformen sind möglich*
- *Es kann mehrere Architectures für eine Entity geben*

Implementierung AND

```
architecture rtl1 of my_and is
begin
   c <= a and b;
end rtl1;


architecture rtl2 of my_and is
begin
   c <= '1' when a = '1' and b = '1' else
        '0';
end rtl2;
```

Unterschied?

Wertetabelle rtl1

b \ a	U	X	0	1	Z	W	L	H	-
U	U	U	0	U	U	U	0	U	U
X	U	X	0	X	X	X	0	X	X
0	0	0	0	0	0	0	0	0	0
1	U	X	0	1	X	X	0	1	X
Z	U	X	0	X	X	X	0	X	X
W	U	X	0	X	X	X	0	X	X
L	0	0	0	0	0	0	0	0	0
H	U	X	0	1	X	X	0	1	X
-	U	X	0	X	X	X	0	X	X

Wertetabelle rtl2

b \ a	U	X	0	1	Z	W	L	H	-
U	0	0	0	0	0	0	0	0	0
X	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0
Z	0	0	0	0	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0
-	0	0	0	0	0	0	0	0	0

Operationen auf std_(u)logic

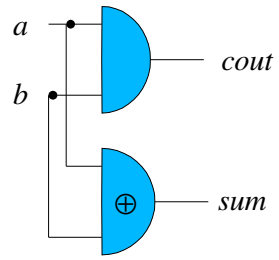
- *and*
- *or*
- *not*
- *nand*
- *nor*
- *xor*
- *xnor*

Wertetabelle NOT

in	
U	U
X	X
0	1
1	0
Z	X
W	X
L	1
H	0
-	X

Beispiel: Halbaddierer

- *Strukturell:*



- *Wertetabelle:*

cout	0	1
0	0	0
1	0	1

sum	0	1
0	0	1
1	1	0

Implementierung Halbaddierer

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

entity ha is
   port(a, b  : in  std_logic;
        sum   : out std_logic;
        cout  : out std_logic);
end ha;

architecture rtl of ha is
begin
   sum  <= a xor b;
   cout <= a and b;
end rtl;
```


Implementierung Volladdierer

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

entity fa is
   port(a, b, cin : in  std_logic;
        sum, cout : out std_logic);
end fa;

architecture rtl_inefficient of fa is
begin
   sum  <= a xor b xor cin;
   cout <= (a and b) or (a and cin) or
           (b and cin);
end rtl_inefficient;
```

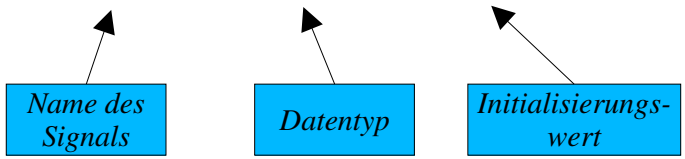

Alternative Implementierung

```
architecture rtl of fa is
signal tmp : std_logic;
begin
   tmp  <= a xor b;
   sum  <= tmp xor cin;
   cout <= (a and b) or (tmp and cin);
end rtl;
```


Signal Declaration

- *Beispiele:*

```
signal s    : std_logic;
signal bus  : std_logic_vector(31 downto 0);
signal u, v : std_logic := '0';
```



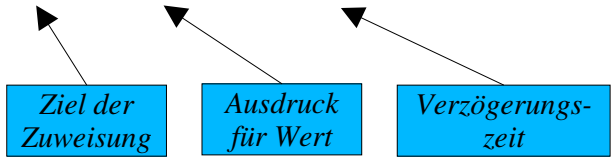
Signal Declaration und Synthese

- *Initialisierungswerte für Signale sind nicht synthetisierbar.*
- *Deshalb sollten alle Signale über ein Reset-Signal initialisiert werden.*

Signal Assignment

- *Beispiele:*

```
out <= a and b;
out <= a and b after 5ns;
out <= '0' after 3ns, '1' after 5ns;
```



- *Verzögerungszeiten sind nicht synthetisierbar!*

Constant Declaration

- *Beispiele:*

```
constant cu : std_logic := 'U';
constant bus_width : integer := 32;
```

- *Verbessern die Les- und Wartbarkeit*
- *Erhöhen die Flexibilität*