Karlsruher Institut für Technologie

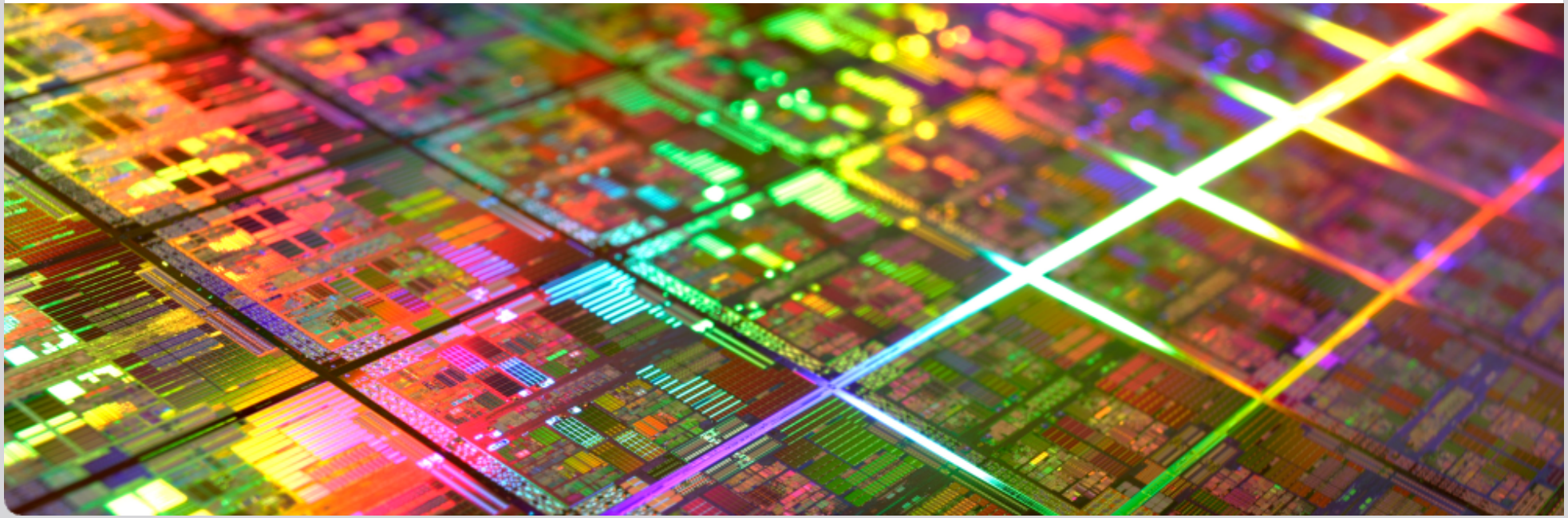
Zentralübung Rechnerstrukturen im SS 2013

Fragen des Rechnerentwurfs: Fertigung und Hardwareentwurf

Oliver Mattes, Prof. Dr. Wolfgang Karl

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

2. Mai 2013



Überblick

Karlsruher Institut für Technologie

Chip-Fertigung:

- Definitionen
- Aufgabe 1: Fertigungskosten

Schaltungsentwurf mit VHDL:

- Einführung in VHDL
- Aufgabe 2: Signale und boolsche Funktionen
- Aufgabe 3: Verhaltensbeschreibung
- Aufgabe 4: VHDL-Entwurfsprozess – DFT
- Aufgabe 5: VHDL-Entwurfsprozess – Zähler

Chip-Fertigung

- Fertigung auf Wafern
  - Größe/Durchmesser des Wafers $\Rightarrow$ Grundfläche
- Heraustrennen der einzelnen Chip-Plättchen (**Die**)
  - Fläche/Form des Dies $\Rightarrow$ Dies per Wafer

- Endkunde erhält:
  - **Chip im Gehäuse**
    - Test des Dies
    - Packaging: Einsetzen des Dies in Gehäuse (Bonding)
    - Finaler Test
  - **Die** für sog. *bond-out*
    - Die wird vom Kunden in Schaltung/Platine direkt integriert
    - Kein Packaging
    - End-Prüfung im Rahmen des Gerätetests

Chip-Fertigung: Kenngrößen

Karlsruher Institut für Technologie

Kenngrößen des Wafers

- $cost_{wafer}$: Fertigungskosten des rohen Wafers (Siliziumscheibe)
- d_{wafer}: Größe/Durchmesser, liefert Fläche $a_{wafer} = \pi(d_{wafer}/2)^2$
- $yield_{wafer}$: Ausbeute („gute" Wafer)

Kenngrößen des Dies

- **Dies per Wafer** (dpw):
  Die-Fläche a_{die} und Wafer-Area a_{wafer} korreliert

$$dpw = A - B = \frac{\pi * (d_{wafer}/2)^2}{a_{die}} - \frac{\pi * d_{wafer}}{\sqrt{2 * a_{die}}}$$

A: theoretisches Maximum
B: Verschnitt

Chip-Fertigung: Kenngrößen (forts.)

Karlsruher Institut für Technologie

Kenngrößen des Dies

- $yield_{die}$: Ausbeute, funktionsfähige Dies
- Fehlerquote (defects per unit area, *dpua*)
- Technologiekonstante (α, Maß für Komplexität bzw. Fertigungstechnologie)

$$yield_{die} = yield_{wafer} * \left(1 + \frac{dpua * a_{die}}{\alpha}\right)^{-\alpha}$$

- Fertigungskosten pro Die

$$cost_{die} = \frac{cost_{wafer}}{dpw * yield_{die}}$$

Chip-Fertigung: Kenngrößen (forts.)

Karlsruher Institut für Technologie

Test und Assemblierung

- Kosten
  - Die-Test: $cost_{die-test}$
  - Packaging: $cost_{packaging}$
- Packaging-Kosten beinhalten zusätzliche Test-Kosten (IC-Test, Endkontrolle)
- Endausbeute: $yield_{final}$

Gesamtkosten

- Pro integriertem Schaltkreis (IC):

$$cost_{IC} = \frac{cost_{die} + cost_{die-test} + cost_{packaging}}{yield_{final}}$$

Aufgabe 1

Eine Wafer-Fertigungsanlage soll von 200 mm- auf 300 mm- Wafer umgestellt werden. Der Fertigungsprozess wird hierbei nicht verändert, der zugehörige Technologiefaktor α sei 2, die Fehlerquote (*defects per unit area*) betrage $0,2/cm^2$ und die Wafer-Ausbeute (*yield*) betrage 80%. Der zu fertigende Die habe eine Fläche von $a_{die} = 4,5\,cm^2$.

a) Berechnen Sie für beide Wafergrößen die erzielbare Anzahl von Dies pro Wafer.

$$dpw = \frac{\pi * \left(d_{wafer} * \frac{1}{2}\right)^2}{a_{die}} - \frac{\pi * d_{wafer}}{\sqrt{2 * a_{die}}}$$

$$dpw_{200} = \frac{\pi * \left(20cm * \frac{1}{2}\right)^2}{4{,}5cm^2} - \frac{\pi * 20\,cm}{\sqrt{2 * 4{,}5\,cm^2}}$$

$$= \pi * \left(\frac{10^2}{4{,}5} - \frac{20}{\sqrt{9}}\right) = \pi * \frac{200-60}{9} = \frac{140}{9}\pi \quad (\approx 48 Dies)$$

$$dpw_{300} = \frac{\pi * \left(30cm * \frac{1}{2}\right)^2}{4.5cm^2} - \frac{\pi * 30cm}{\sqrt{2 * 4.5cm^2}}$$

$$= \pi * \left(\frac{450}{9} - \frac{30}{\sqrt{9}}\right) = \pi * (50 - 10) = 40\pi \quad (\approx 125 Dies)$$

Aufgabe 1 (forts.)

Eine Wafer-Fertigungsanlage soll von 200 mm- auf 300 mm-Wafer umgestellt werden. Der Fertigungsprozess wird hierbei nicht verändert, der zugehörige Technologiefaktor α sei 2, die Fehlerquote (*defects per unit area*) betrage $0,2/cm^2$ und die Wafer-Ausbeute (*yield*) betrage 80%. Der zu fertigende Die habe eine Fläche von $a_{die} = 4,5\,cm^2$.

b) Errechnen Sie den Die-Yield für die gegebenen Parameter.

$$yield_{die} = yield_{wafer} * \left(1 + \frac{dpua * a_{die}}{\alpha}\right)^{-\alpha}$$

$$yield_{die} = 0,8 * \left(1 + \frac{0,2 * 4,5}{2}\right)^{-2} = 0,8 * 0,476 = 0,38$$

Aufgabe 1 (forts.)

c) Errechnen Sie die Kosten pro Die für 200 mm und 300 mm-Technologie unter der Annahme, dass ein 200 mm-Wafer 150 Euro kostet und ein 300 mm-Wafer 300 Euro.

$$cost_{die} = \frac{cost_{wafer}}{dpw * yield_{die}}$$

Berechnet: $dpw_{200} = 48, dpw_{300} = 125, yield_{die} = 0,38$

$$cost_{200} = \frac{150\,€}{48*0,38} = 8,22\,€$$

$$cost_{300} = \frac{300\,€}{125*0,38} = 6,32\,€$$

Aufgabe 1 (forts.)

d) Berechnen Sie basierend auf den errechneten Werten der vorherigen Aufgabenteile die durch die Umstellung auf 300 mm-Wafer erzielte Kostenreduzierung pro IC. Die Kosten für das Packaging pro IC betragen 75 cent, der Kostenanteil für Testen des einzelnen Dies sei 1 Euro und die Gesamtausbeute sei 75 %.

$$cost_{ic} = \frac{cost_{die} + cost_{test} + cost_{pkg}}{yield_{final}}$$

$$cost_{ic200} = \frac{8{,}22\,€ + 1\,€ + 0{,}75\,€}{0{,}75} = \frac{9{,}97\,€ * 4}{3} = 13{,}29\,€$$
$$cost_{ic300} = \frac{6{,}32\,€ + 1\,€ + 0{,}75\,€}{0{,}75} = \frac{8{,}07\,€ * 4}{3} = 10{,}76\,€$$

Einsparung: $13{,}29\,€ - 10{,}76\,€ = 2{,}53\,€$

Kostensenkung um:
$1 - \frac{10{,}76\,€}{13{,}29\,€} = 1 - 0{,}81 = 100\,\% - 81\,\% = 19\%$

Überblick

Chip-Fertigung:

- Definitionen
- Aufgabe 1: Fertigungskosten

Schaltungsentwurf mit VHDL:

- Einführung in VHDL
- Aufgabe 2: Signale und boolsche Funktionen
- Aufgabe 3: Verhaltensbeschreibung
- Aufgabe 4: VHDL-Entwurfsprozess – DFT
- Aufgabe 5: VHDL-Entwurfsprozess – Zähler

VHDL – Einführung

Karlsruher Institut für Technologie

VHDL

- **VHDL** – **V**HSIC **HDL** – Very-High-Speed Integrated Circuits Hardware Description Language
- Beschreibungssprache, keine Programmiersprache
- Vom US-Verteidigungsministerium zur Dokumentation und Simulation von Schaltungen / ASICs ins Leben gerufen
- (HW-)Synthese erst später
- Syntax ähnlich der Programmiersprache Ada

- **Simulations- und Syntheseprogramme unterstützen nicht unbedingt den kompletten Sprachumfang**

VHDL – Hauptbestandteile

Karlsruher Institut für Technologie

Entity

- Definition der Schnittstelle eines Hardwaremoduls

Architecture

- **Funktionale Verhaltensbeschreibung**
- **Strukturelle Beschreibung** eines Moduls
- Mischung beider Beschreibungsarten

Configuration

Zuweisung einer Architecture zur verwendeten Entity

VHDL – Entity

Entity

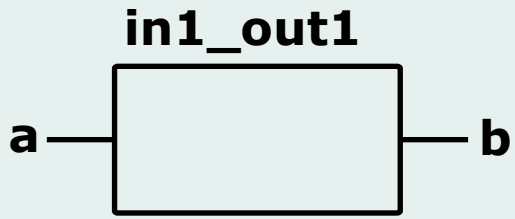
- **Definition der Schnittstelle** („Leere Hülle“)

$\Rightarrow$ Ein- und Ausgänge (Ports) eines Hardwaremoduls

- Festlegung der Datentypen der Ein- und Ausgänge

Entity

```
-- Ich bin ein Kommentar
entity in1_out1 is
  port (
   a : in bit;
   b : out bit
  );
end in1_out1;
```



VHDL – Signale und Variablen

Karlsruher Institut für Technologie

Signale

- Verbindung zwischen verschiedenen Komponenten
- Signale speichern (puffern) auch Werte
- Signale haben einen Datentyp
- Deklaration:

```
signal a, b, out : bit;
```

- Wertzuweisung durch <=

```
out <= '1';
```

- **Achtung: Keine mehrfachen Zuweisungen!**
- Wertzuweisung erst gültig nach der Abarbeitung aller Operationen des umfassenden Blocks!

VHDL – Signale und Variablen

Karlsruher Institut für Technologie

Variablen

- Werte innerhalb eines Prozesses
- Variablen haben einen Datentyp
- Deklaration:

```
variable state : bit;
```

- Wertzuweisung durch `:=`

```
state := not state;
```

- Wertzuweisung erfolgt sofort
- Deklaration mit Standardwert:

```
variable state : bit := '1';
```

VHDL – Prozess

Prozess

- Haupteinheit einer Verhaltensbeschreibung in VHDL
- **Sensitivity List**: Signale, bei deren Änderung der Prozess ausgeführt wird
- **Mehrere Prozesse werden gleichzeitig abgearbeitet**
- Rückhalten der Zuweisungen bis Blockende

Beispiel

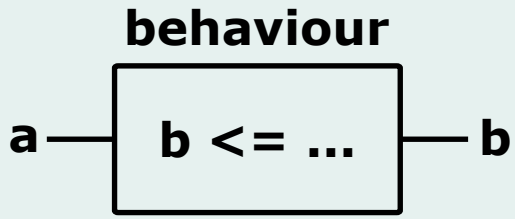
```
invertieren : process (invert)
  variable state : bit := '1';
begin
  if invert'event then
    state := not state;
  end if;
  out <= state;
end process invertieren;
```

VHDL – Architecture

Architecture

- „Füllung“ einer Entity mit Inhalt
- Inhalt, d.h. Logik bzw. Aufbau der Schaltung
- Mehrere Architekturen können für eine Entiy definiert werden

⇒ Auswahl mit Hilfe einer Configuration



- Verschiedene Möglichkeiten:
  - **Funktionale Verhaltensbeschreibung**
  - **Strukturelle Beschreibung** eines Moduls
  - Mischung beider Beschreibungsarten

VHDL – Architecture

Funktionale Verhaltensbeschreibung

- Beschreibung der Funktionalität einer Schaltung

- Die interne Struktur wird abstrahiert
- Verhaltensbeschreibung vergleichbar mit einem Programm
- In einfachen Fällen durch eine direkte Funktion

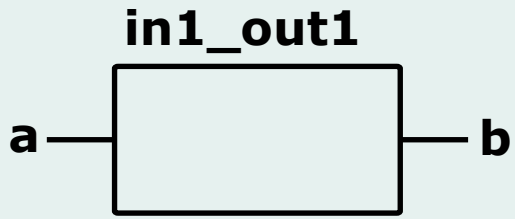
- Wird oft „Behaviour“ genannt.

- Hauptbestandteil einer Verhaltensbeschreibung ist der Prozess
- Funktionales Verhalten durch **nebenläufige Anweisungen**

VHDL – Beispiel

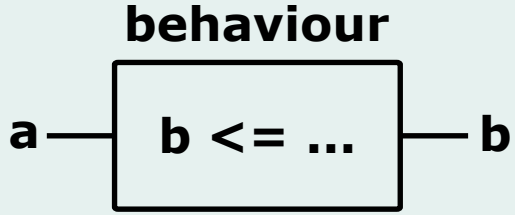
Entity

```
entity in1_out1 is
  port (
   a : in bit;
   b : out bit
  );
end in1_out1;
```



Architecture – Verhaltensbeschreibung

```
architecture behaviour of in1_out1 is
begin
  invertieren : process(a)
  begin
    if a = '1' then
      b <= '0';
    else
      b <= '1';
    end if;
  end process invertieren;
end behaviour;
```



VHDL – Architecture

Karlsruher Institut für Technologie

Strukturelle Beschreibung

- Zusammensetzen verschiedener Untermodule zu einer Schaltung
- Beschreibung einer Schaltung durch Aufteilung in verschiedene Untermodule

- Wird oft „Structure“ genannt.

VHDL – Architecture

Karlsruher Institut für Technologie

Strukturelle Beschreibung

Verwendung:

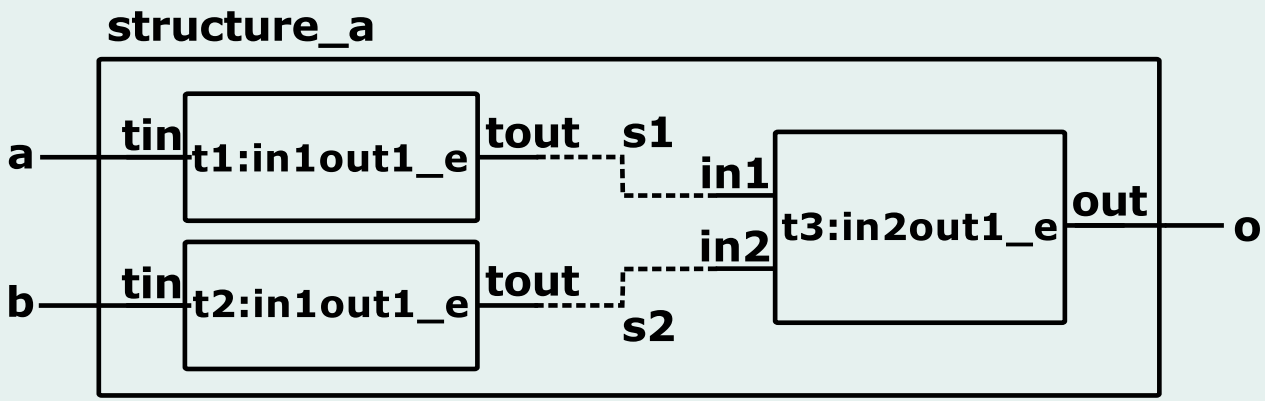
1. **Komponentendeklaration**
   - Deklaration der Schnittstelle der Untermodule
   - $\Rightarrow$ Wiederholung der Entity des Untermoduls
   - Durch die Deklaration wird kein Untermodul erzeugt!
2. **Instantiierung** der Untermodule
   - Erzeugung eines oderer mehrerer Untermoduls des angegebenen Typs
   - Mehrere Instanzen, d.h. mehrere Untermoduls vom selben Typ möglich

- Verbindung der Untermodule durch Signale
  - Angabe der Verbindungen mit Hilfe der „port map“

VHDL – Architecture – Beispiel

Architecture – Strukturelle Beschreibung

```
architecture structure_a of entity_name is
  component in1out1_e
    port (tin : in bit; tout : out bit);
  end component;
  component in2out1_e
    port (in1 : in bit; in2 : in bit; out : out bit);
  end component;
  signal s1, s2 : bit;
begin
  t1 : in1out1_e port map (a, s1);
  t2 : in1out1_e port map (b, s2);
  t3 : in2out1_e port map (out => o, in1 => s1, in2 => s2);
end structure_a;
```



VHDL – Configuration

Karlsruher Institut für Technologie

Configuration

- Auswahl der gewünschten Architekturbeschreibung für eine Entity (d.h. Auswahl des internen Aufbaus)
- Auswahl der zu verwendenden Beschreibungen für einzelne Instanzen in struktuellen Beschreibungen
- Auswahl der gewünschten Architekturbeschreibung für einzelne/alle Instanzen einer Entity
- Festlegung des internen Aufbaus bei mehreren Möglichkeiten
- Verbindung von Signalen und Ports,. . .

Möglichkeiten zur Konfiguration

- Zusätzliche Schaltungseinheit „configuration“
- Komponentenkonfiguration in der Architekturbeschreibung

Aufgabe 2

Aufgabe 2 – Signale und boolesche Funktionen

Erstellen Sie je eine VHDL-Beschreibung der XOR-Funktion mittels

a) Bibliotheksaufruf
b) Boolescher Beschreibung
c) Wertetabelle
d) Beschreibung der Funktion

Aufgabe 2

a) Bibliotheksaufruf

Einladen benötigter Bibliotheken:

```
library IEEE;
use IEEE.std_logic_1164.all;


c <= a XOR b;
```

b) Boolesche Beschreibung

XOR : $(A \wedge \neg B) \vee (\neg A \wedge B)$

```
c <= (a and not(b)) or (not(a) and b);
```

Aufgabe 2

c) Wertetabelle

a	b	c
0	0	0
0	1	1
1	0	1
1	1	0

```
c <='0' when a='0' and b='0' else
     '1' when a='0' and b='1' else
     '1' when a='1' and b='0' else
     '0' when a='1' and b='1';
```

Aufgabe 2

d) Beschreibung der Funktion

```
c <= '0' when a=b else '1';
```

bzw.

```
if ((a='1' and b='1') or (a='0' and b='0')) then
  c <= '0';
else
  c <= '1';
end if;
```

Aufgabe 3 – Verhaltensbeschreibung

Karlsruher Institut für Technologie

Aufgabe 3 – Verhaltensbeschreibung

Eine zu entwickelnde Zählerschaltung soll folgendes Verhalten aufweisen:

- Ein low-aktives Rücksetzsignal löscht den Zähler.
- Über ein Richtungssignal wird bestimmt, ob der Zähler mit der steigenden Flanke eines Taktsignals aufwärts (=0) oder abwärts (=1) zählt.
- Es wird nur gezählt, wenn der Zähler mit einem high-aktiven Aktivierungssignal freigeschaltet ist.
- Der Zähler soll 64 Zählschritte ausführen können.
- Ein low-aktives Freigabesignal entscheidet, ob der Zählerausgang auf einen gemeinsamen Bus gelegt werden soll; bei nicht erfolgter Freigabe werden die Ausgabeleitungen in den Tristate-Zustand geschaltet.

Aufgabe 3 – Verhaltensbeschreibung

Karlsruher Institut für Technologie

a) Erstellen Sie die zugehörige **Schnittstellenbeschreibung** und formulieren Sie die entsprechende Verhaltensbeschreibung in VHDL.

Schnittstellenbeschreibung

- **Entity** beschreibt Ein-/Ausgabeschnittstelle
- Angabe der Schnittstellen über `Port`
- Richtung und Type der Ports
- Pro Modul nur eine Entity erlaubt
- Parametrisierung über `Generic` möglich

Aufgabe 3 – Verhaltensbeschreibung

Karlsruher Institut für Technologie

Lösung: Schnittstellenbeschreibung

```
entity counter is
 port (
  clk, rst_n : in  std_logic;
  direction  : in  std_logic;
  enable     : in  std_logic; --enable circuit
  select_n   : in  std_logic; --read counter value
  value      : out std_logic_vector(5 downto 0)
 );
end entity;
```

Aufgabe 3 – Verhaltensbeschreibung

a) Erstellen Sie die zugehörige Schnittstellenbeschreibung und formulieren Sie die entsprechende **Verhaltensbeschreibung** in VHDL.

Verhaltensbeschreibung

- Implementierung der Funktionalität eines (Teil-)Moduls in einer **Architecture**
- „Berechnung“ der Ausgangssignalwerte anhand der Eingangssignale und des intrinsischen Zustands
- Prozesse zur Bündelung
  - Alle Prozesse laufen prinzipiell parallel
- Nebenläufige / Asynchrone Zuweisungen

Aufgabe 3 – Verhaltensbeschreibung

Karlsruher Institut für Technologie

Lösung: Verhaltensbeschreibung

Problemfall: `value` ist als reines Ausgabesignal deklariert und kann nicht als eigentlicher Zähler verwendet werden. In der Verhaltensbeschreibung muss zusätzlich der eigentliche Zähler als Signal deklariert werden. Die Ausgabe des Zählers erfolgt außerhalb des Prozesses.

```
architecture arch_counter of counter is
  signal count : unsigned(5 downto 0) := "000000"; -- 2^6=64 Zustände
begin

 -- Ausgabe
 value <= std logic vector(count) when ena='0'
          else (others=>'Z');

end arch_counter;
```

Aufgabe 3 – Verhaltensbeschreibung

a) Erstellen Sie die zugehörige Schnittstellenbeschreibung und formulieren Sie die entsprechende **Verhaltensbeschreibung** in VHDL.

VHDL-Prozesse

- Implementierung einer/der (Teil-)Funktionalität
- Reagieren auf Ereignisse in Sensibilitätsliste:
  `process (clk, rst_n)`
- Zustand häufig über Zustandsautomaten gehandhabt
- Verwendung logischer und arithmetischer Operatoren
- if/case/when zur Kontroll- und Datenflusssteuerung

Aufgabe 3 – Verhaltensbeschreibung

Karlsruher Institut für Technologie

Lösung: Verhaltensbeschreibung

Prozess zerfällt in asynchrones Rücksetzen und eigentlichen Zählvorgang.

```
p_counter: process (rst_n, clk),
begin
 -- asynchronous reset
 if rst_n='0' then
  count <= "000000";

 -- counting function
 elsif clk'event and clk='1' then

  -- counter enabled?
  if enable = '1' then
   -- counting direction
   if direction='0' then
    count <= count+1;
   else
    count <= count-1;
   end if;

  end if;

 end if;
end process;
```

Aufgabe 3 – Verhaltensbeschreibung

Karlsruher Institut für Technologie

b) Erweitern Sie die Verhaltensbeschreibung um eine Lösung, bei der ein Überlaufsignal außerhalb des Prozesses erzeugt wird.

Über-/Unterlaufsfunktion

Der Zähler soll um eine Über-/Unterlaufsfunktion ergänzt werden, d.h. beim Wechsel von 63 zu 0 (*Aufwärtszählen*) bzw. 0 zu 63 (*Abwärtszählen*) wird für die Dauer eines Taktes ein Anzeigesignal ausgegeben. Hierbei soll das Rücksetzsignal nicht fälschlicherweise das Überlaufssignal auslösen.
In der Entity sei hierzu das zusätzliche Signal `ovl` vom Typ `std_logic` mit Modus `out` deklariert.

Aufgabe 3 – Verhaltensbeschreibung

Karlsruher Institut für Technologie

Über-/Unterlaufsfunktion

Da eine Abfrage auf Zählerstand 0 auch im Reset-Fall auslösen würde, ist hier eine Lösung zu finden, um festzustellen, ob tatsächlich ein Über-/Unterlauf stattfand. Hierzu wird das höchstwertige Bit des Zählers gespeichert und in den Test auf Zählerstand 0 miteinbezogen. Im Unterschied zum nachfolgenden Aufgabenteil kann hier direkt auf den entsprechenden Zählerstand verglichen werden. Bezüglich der Komplexität der Abfrage ändert sich nichts.

```
architecture arch_counter_ovl2 of counter is
  signal count : unsigned(5 downto 0) := "000000";
  signal store : std_logic; -- Zustandsspeicher
begin
```

Aufgabe 3 – Verhaltensbeschreibung

Karlsruher Institut für Technologie

```
p_counter: process(rst_n, clk)
begin
  -- asynchrones Rücksetzen
  if rst_n='0' then
    count <= "000000";
    store <= '0';

  -- Zählfunktion
  elsif clk'event and clk='1' then

    -- Bit 5 des Zählers merken
    store <= count(5);

    -- Zähler selektiert?
    if enable='1' then
```

Aufgabe 3 – Verhaltensbeschreibung

Karlsruher Institut für Technologie

```
        -- Zählrichtung
        if direction='0' then
          count <= count+1;
        else
          count <= count-1;
        end if;

      end if;

    end if;
  end process;
```

Aufgabe 3 – Verhaltensbeschreibung

Karlsruher Institut für Technologie

```
  -- Über/Unterlauf?
  ovl<='1' when count="000000" and store='1' and direction='0'
       else '1' when count="111111" and direction='1'
       else '0';

  -- Ausgabe
  value <= std_logic_vector(count) when select_n='0'
           else (others=>'Z');

end arch_counter_ovl2;
```

Karlsruher Institut für Technologie

Aufgabe 4: VHDL-Entwurfsprozess I – DFT

- Realisierung einer Fourier-Transformation
- Durchführung von
  - Datenverfeinerung
  - Strukturverfeinerung
  - Verhaltensverfeinerung

Aufgabe 4 - VHDL-Entwurfsprozess I - DFT

Karlsruher Institut für Technologie

Aufgabe 4a)

Datenverfeinerung:

- Realisierung abstrakter Datentypen durch einfachere Datentypen
- Synthese: Binärer Datentyp bzw. erweiterter binärer Datentyp (incl. Tri-state-Zustand, `std_logic`)

Datenverfeinerung und Schnittstellenbeschreibung

- Stream kann nicht direkt modelliert werden
- Passende Schnittstelle:
  Daten, Gültigkeitsanzeige, Aufnahmebereitschaft, Stream-Ende

Aufgabe 4 - VHDL-Entwurfsprozess I - DFT

Karlsruher Institut für Technologie

Lösung: Schnittstellenbeschreibung

```
entity DFT_top is
  generic (
   C_DATA_SIZE : integer := 16 -- Parametrisierbare Datengröße
  );
  port (
    clk       : in  std_logic;
    rst_n     : in  std_logic; -- low-aktives Rücksetzen
    en        : in  std_logic; -- Aktivierungssignal

    -- Eingangs-Stream
    data      : in  std_logic_vector(C_DATA_SIZE-1 downto 0);
    valid     : in  std_logic;
    not_full  : out std_logic;
    eos       : in  std_logic
  );
end entity;
```

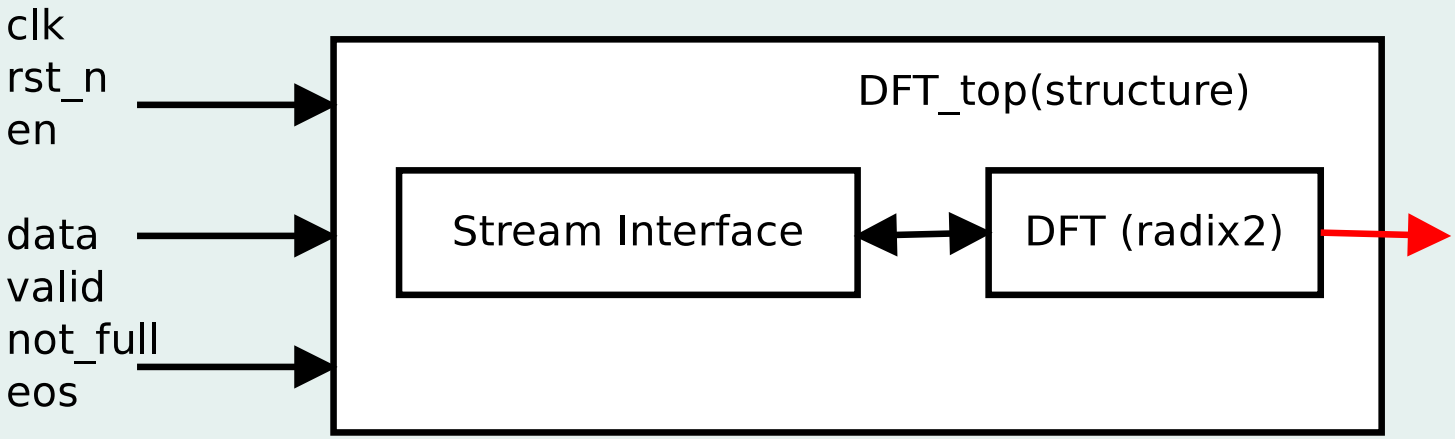
Aufgabe 4 - VHDL-Entwurfsprozess I - DFT

Aufgabe 4b)

Strukturverfeinerung:

Realisierung einer spezifizierten Funktion durch Verschaltung von Komponenten mit einfacherer Funktionalität

Lösung: Architektur



(roter Pfeil: sinnvollerweise Ausgabe der transformierten Daten)

Aufgabe 4 - VHDL-Entwurfsprozess I - DFT

Karlsruher Institut für Technologie

```
architecture structure_top of DFT_top is

  component stream_interface
    generic ( C_DATA_SIZE : integer := 16 );
    port (
      clk           : in std_logic;
      rst           : in std_logic;
      -- pass data to fourier instance
      data_out      : out std_logic_vector(C_DATA_SIZE-1 downto 0);
      valid_out     : out std_logic;
      not_full_in   : in std_logic;
      eos_out       : out std_logic;
      -- new data from outside
      data_in       : in std_logic_vector(C_DATA_SIZE-1 downto 0);
      valid_in      : in std_logic;
      not_full_out  : out std_logic;
      eos_in        : in std_logic
    );
  end component;
```

Aufgabe 4 - VHDL-Entwurfsprozess I - DFT

Karlsruher Institut für Technologie

```
component DFT
  generic ( C_DATA_SIZE : integer := 16 );
  port (
    clk      : in std_logic;
    rst      : in std_logic;
    data     : in std_logic_vector(C_DATA_SIZE-1 downto 0);
    valid    : in std_logic;
    not_full : out std_logic;
    eos      : in std_logic
  );
end component
```

Aufgabe 4 - VHDL-Entwurfsprozess I - DFT

Karlsruher Institut für Technologie

```
  signal data_s2f     : std_logic_vector(C_DATA_SIZE-1 downto 0);
  signal valid_s2f    : std_logic;
  signal not_full_f2s : std_logic;
  signal eos_s2f      : std_logic;

  signal not_full_i   : std_logic;

  signal rst          : std_logic;

begin

  -- buffer output ports
  not_full <= not_full_i;

  rst      <= not rst_n;
```

Aufgabe 4 - VHDL-Entwurfsprozess I - DFT

Karlsruher Institut für Technologie

```
stream_instance : stream_interface
 generic map ( C_DATA_SIZE => 16 )
 port map (
  clk          => clk;
  rst          => rst;

  --pass data to fourier instance
  data_out     => data_s2f;
  valid_out    => valid_s2f;
  not_full_in  => not_full_f2s;
  eos_out      => eos_s2f;

  --new data from outside
  data_in      => data;
  valid_in     => valid;
  not_full_out => not_full_i;
  eos_in       => eos
);
```

```
fourier_instance : DFT
 generic map (C_DATA_SIZE => 16)
 port map (
  clk      => clk;
  rst      => rst;
  data     => data_s2f;
  valid    => valid_s2f;
  not_full => not_full_f2s;
  eos      => eos_s2f
);

end structure_top;
```

Aufgabe 4 - VHDL-Entwurfsprozess I - DFT

Karlsruher Institut für Technologie

Aufgabe 4c)

Verhaltensverfeinerung:

- Detailliertes Ausarbeiten der gewünschten Funktionalität
- Ersetzen von Black-Boxes

Behandlung/Darstellung der komplexen Einheitswurzel

- Einheitswurzel: $e^{-2\pi ik/N}$
- $e^{ix} = \cos x + i \cdot \sin x$
- N und mögliche k bekannt $\Rightarrow$ Wertetabelle anlegen für die Sinus-Werte $\sin(2k\pi/N) \quad \forall\ 0 <= k < N$
- ```
  type rom_type is array(integer range <>) of float;
  sine_rom : rom_type(0 to N-1) := {0.0, 0.383,
                                    0.707, 0.924, 1.0,
                                    0.923  ... -0.383};
  ```
- Cosinus-Werte: Andere Indizierung der Tabelle
  ```

Aufgabe 4 - VHDL-Entwurfsprozess I - DFT

Karlsruher Institut für Technologie

Aufgabe 4d)

Configuration:

- Zuordnung einer Architektur zu einer Schnittstelle
- Festlegung verwendeter Architecture zu verwendeter Komponente
- Konfiguration, z.B. von Signalbreiten

Aufgabe 4 - VHDL-Entwurfsprozess I - DFT

Karlsruher Institut für Technologie

Lösung: Radix-2-Variante

- Auswahl einer passenden Architektur über Konfiguration:

```
configuration cfg of DFT_top is:
  -- for which architecture?
  for structure
    -- for which instance/component?
    for all : DFT
      -- architecture "radix2" of DFT
      use entity work.DFT(radix2);
    end for;
  end for;
end cfg;
```

Aufgabe 5 - VHDL-Entwurfsprozess II

Karlsruher Institut für Technologie

In einer VHDL-Beschreibung sei ein Prozess wie folgt beschrieben.

```
architecture behaviour of counter is
  signal count : unsigned(7 downto 0);
begin

process(clk)
begin
  if clk'event and clk='1' then
    count <= count+1;
    if count=X"ff" then
      flag <= '1';
    else
      flag <= '0';
    end if;
  end if;
end process;
end behaviour;
```

Aufgabe 5 - VHDL-Entwurfsprozess II

a) Bei der Simulation dieses Prozesses erhalten Sie immer den Wert `"UUUUUUUU"` für das Signal `count`. Synthetisiert in Hardware beobachten Sie jedoch wie erwartet eine Aufwärtszählfunktion.

- Nennen Sie die Ursache für das in der Simulation beobachtete Verhalten und erklären Sie, weswegen die Zählfunktion hier nicht sichtbar wird.

⇒ `count` ist undefiniert , der Wert `count+1` („undefiniert+1“) ist daher ebenfalls undefiniert.

⇒ Initialisierung mit Standardwert:

```
architecture behaviour of counter is
  signal count : unsigned(7 downto 0) := "00000000";
begin
...
```

Aufgabe 5 - VHDL-Entwurfsprozess II

a) Bei der Simulation dieses Prozesses erhalten Sie immer den Wert `"UUUUUUUU"` für das Signal `count`. Synthetisiert in Hardware beobachten Sie jedoch wie erwartet eine Aufwärtszählfunktion.

- Ändern Sie die Prozessbeschreibung so ab, dass der Wert des Zählers zurückgesetzt werden kann.

⇒ Zurücksetzen des Zählers mittels Reset:

```
process(rst, clk)
begin
  if rst='1'
    then count<="00000000";
  elseif clk'event and clk='1' then
    count <= count+1;
    ...
  end if;
end process;
```

Aufgabe 5 - VHDL-Entwurfsprozess II

b) Das `flag`-Signal (vom Typ `bit`) soll den Zählerstand `0xff` anzeigen, d.h. zum Zeitpunkt `count=X'ff'` für eine Taktperiode den Wert `1` annehmen, sonst `0`.

- Bei welchen tatsächlichen Zählerstand beobachten Sie beim gegebenen Codefragment in Simulation und Synthese den Zustand `flag='1'`? Warum ist dies so?

$\Rightarrow$ Die Zählerinkrementierung wird erst mit einer Verzögerung von einem Taktzyklus sichtbar, darum wird das Signal `flag` tatsächlich den Zählerstand `0x00` signalisieren.

Aufgabe 5 - VHDL-Entwurfsprozess II

b) Das `flag`-Signal (vom Typ `bit`) soll den Zählerstand `0xff` anzeigen, d.h. zum Zeitpunkt `count=X'ff'` für eine Taktperiode den Wert `1` annehmen, sonst `0`.

- Das `flag`-Signal soll nicht synchron innerhalb des Prozesses sondern nebenläufig außerhalb erzeugt werden. Wie lautet die VHDL-Zuweisung hierfür bzw. was muß am Quelltext geändert werden?

$\Rightarrow$ `flag<='1' when count=X'ff' else '0';`

Ausblick

Übung #2

- Low-Power-Entwurf / Leistungsaufnahme
- Leistungsbewertung von Rechensystemen
- (Fehlertoleranz)

Kontakt

Karlsruher Institut für Technologie

Übungsleiter

- Martin Schindewolf

  `schindewolf@kit.edu`

  0721/608-46048

  Raum B2-314.1

- Oliver Mattes

  `mattes@kit.edu`

  0721/608-48066

  Raum B2-308.2

Webseite

https://capp.itec.kit.edu/teaching/rs/

Karlsruher Institut für Technologie

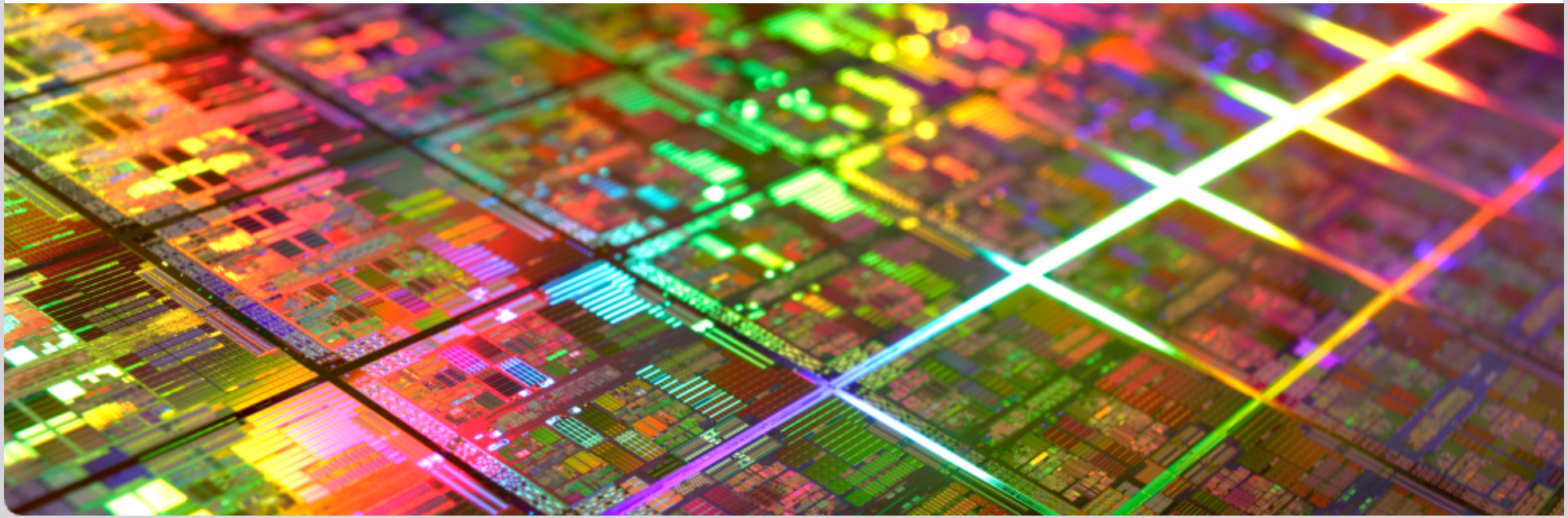
Fragen?

Karlsruher Institut für Technologie

Zentralübung Rechnerstrukturen im SS 2013

Fragen des Rechnerentwurfs: Fertigung und Hardwareentwurf

Oliver Mattes, Prof. Dr. Wolfgang Karl

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

2. Mai 2013