Karlsruher Institut für Technologie

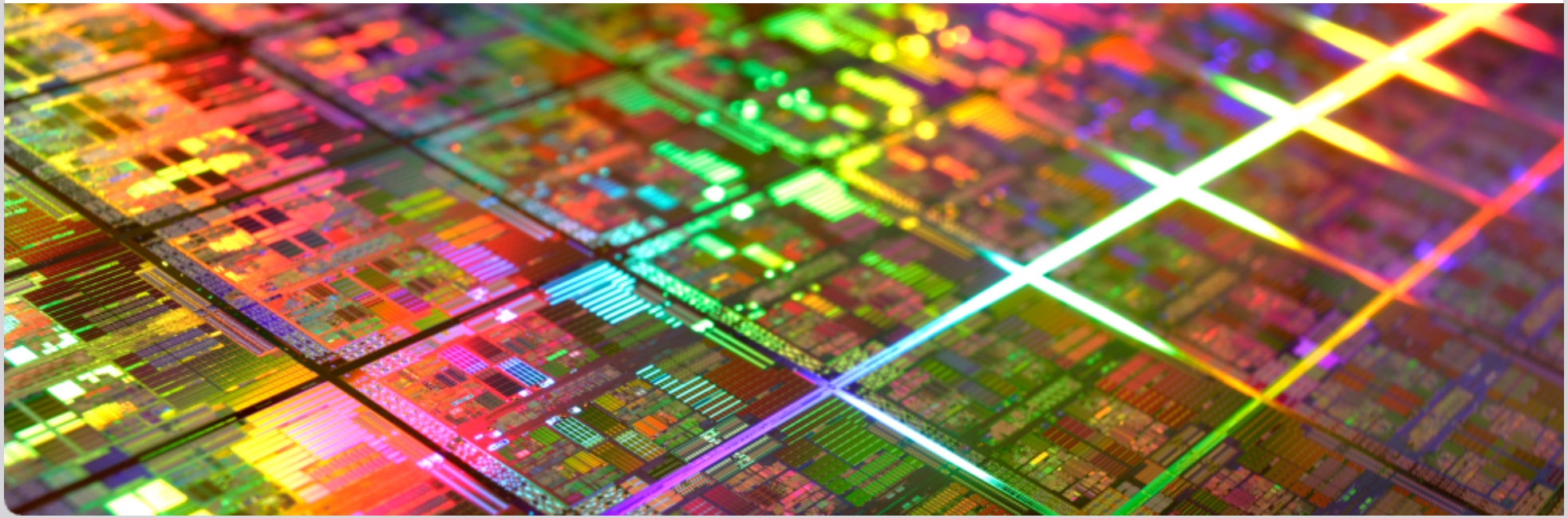
Zentralübung Rechnerstrukturen im SS 2012

Fragen des Rechnerentwurfs: Fertigung und Hardwareentwurf

David Kramer, Prof. Dr. Wolfgang Karl

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

26. April 2012



Organisatorisches

Karlsruher Institut für Technologie

Klausur

- Klausurtermin: 09.08.2012, 14:00 Uhr
- Hörsäle: Gerthsen, Daimler, Benz, Redtenbacher
- Anmeldebeginn: 31.05.2012
- Anmeldeende: 06.08.2012
- Abmeldeende: 07.08.2012

- Es sind **keine** Hilfsmittel zugelassen!

Inhalt

1. Chip-Fertigung
2. Schaltungsentwurf mit VHDL

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}
- $yield_{wafer}$: 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)

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

- 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 200mm- auf 300mm- 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 $a_{die} = 4.5cm^2$.

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

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

$$dpw_{200} = \frac{\pi * (20cm * \frac{1}{2})^2}{4.5cm^2} - \frac{\pi * 20cm}{\sqrt{2 * 4.5cm^2}}$$

$$= \pi * (\frac{10^2}{4.5} - \frac{20}{\sqrt{9}}) = \pi * \frac{200-60}{9} = \frac{140}{9}\pi (\approx 48 Dies)$$

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

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

Aufgabe 1

Eine Wafer-Fertigungsanlage soll von 200mm- auf 300mm- 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 $a_{die} = 4.5cm^2$.

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

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

$$dpw_{200} = \frac{\pi * (20cm * \frac{1}{2})^2}{4.5cm^2} - \frac{\pi * 20cm}{\sqrt{2 * 4.5cm^2}}$$

$$= \pi * (\frac{10^2}{4.5} - \frac{20}{\sqrt{9}}) = \pi * \frac{200-60}{9} = \frac{140}{9}\pi (\approx 48 Dies)$$

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

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

Aufgabe 1

Karlsruher Institut für Technologie

Eine Wafer-Fertigungsanlage soll von 200mm- auf 300mm- 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 $a_{die} = 4.5cm^2$.

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

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

$$dpw_{200} = \frac{\pi * (20cm * \frac{1}{2})^2}{4.5cm^2} - \frac{\pi * 20cm}{\sqrt{2 * 4.5cm^2}}$$
$$= \pi * (\frac{10^2}{4.5} - \frac{20}{\sqrt{9}}) = \pi * \frac{200-60}{9} = \frac{140}{9}\pi (\approx 48 Dies)$$

$$dpw_{300} = \frac{\pi * (30cm * \frac{1}{2})^2}{4.5cm^2} - \frac{\pi * 30cm}{\sqrt{2 * 4.5cm^2}}$$
$$= \pi * (\frac{450}{9} - \frac{30}{\sqrt{9}}) = \pi * (50 - 10) = 40\pi (\approx 125 Dies)$$

Aufgabe 1 (forts.)

Karlsruher Institut für Technologie

Eine Wafer-Fertigungsanlage soll von 200mm- auf 300mm- 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 $a_{die} = 4.5cm^2$.

2. Errechnen Sie den Die-Yield für die gegebenen Parameter.

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

$$yield_{die} = 0.8 * (1 + \frac{0.2 * 4.5}{2})^{-2} = 0.8 * 0.476 = 0.38$$

Aufgabe 1 (forts.)

Karlsruher Institut für Technologie

Eine Wafer-Fertigungsanlage soll von 200mm- auf 300mm- 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 $a_{die} = 4.5cm^2$.

2. Errechnen Sie den Die-Yield für die gegebenen Parameter.

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

$$yield_{die} = 0.8 * (1 + \frac{0.2 * 4.5}{2})^{-2} = 0.8 * 0.476 = 0.38$$

Aufgabe 1 (forts.)

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

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

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

$$cost_{200} = \frac{150}{48 * 0.38} = 8.22 Euro$$

$$cost_{300} = \frac{300}{125 * 0.38} = 6.32 Euro$$

Aufgabe 1 (forts.)

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

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

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

$$cost_{200} = \frac{150}{48 * 0.38} = 8.22 Euro$$

$$cost_{300} = \frac{300}{125 * 0.38} = 6.32 Euro$$

Aufgabe 1 (forts.)

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

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

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

$$cost_{200} = \frac{150}{48 * 0.38} = 8.22 Euro$$

$$cost_{300} = \frac{300}{125 * 0.38} = 6.32 Euro$$

Aufgabe 1 (forts.)

4. Berechnen Sie basierend auf den errechneten Werten der vorherigen Aufgabenteile die durch die Umstellung auf 300mm-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%.

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

$$cost_{ic200} = \frac{8.22+1+0.75}{0.75} = \frac{9.97*4}{3} = 13.29 Euro$$

$$cost_{ic300} = \frac{6.32+1+0.75}{0.75} = \frac{8.07*4}{3} = 10.76 Euro$$

$$Einsparung: 13.29 - 10.76 = 2.53 Euro$$

$$Kostensenkung\ um$$

$$(1 - \frac{13.29}{10.76}) * 100\% = 100\% - 80.9\% = 19,1\%$$

Aufgabe 1 (forts.)

4. Berechnen Sie basierend auf den errechneten Werten der vorherigen Aufgabenteile die durch die Umstellung auf 300mm-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%.

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

$$cost_{ic200} = \frac{8.22+1+0.75}{0.75} = \frac{9.97*4}{3} = 13.29 Euro$$

$$cost_{ic300} = \frac{6.32+1+0.75}{0.75} = \frac{8.07*4}{3} = 10.76 Euro$$

Einsparung: $13.29 - 10.76 = 2.53 Euro$

Kostensenkung um

$$(1 - \frac{13.29}{10.76}) * 100\% = 100\% - 80.9\% = 19,1\%$$

Aufgabe 1 (forts.)

4. Berechnen Sie basierend auf den errechneten Werten der vorherigen Aufgabenteile die durch die Umstellung auf 300mm-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%.

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

$$cost_{ic200} = \frac{8.22+1+0.75}{0.75} = \frac{9.97*4}{3} = 13.29 Euro$$
$$cost_{ic300} = \frac{6.32+1+0.75}{0.75} = \frac{8.07*4}{3} = 10.76 Euro$$

Einsparung: $13.29 - 10.76 = 2.53 Euro$

Kostensenkung um

$$(1 - \frac{13.29}{10.76}) * 100\% = 100\% - 80.9\% = 19,1\%$$

Inhalt

1. Chip-Fertigung
2. Schaltungsentwurf mit VHDL

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

Einführung in VHDL

- VHDL - VHSIC HDL - Very-High-Speed Integrated Circuits Hardware Description Language
- Hardwarebeschreibungssprache
- Syntax ähnlich der Programmiersprache Ada
- Vom US-Verteidigungsministerium zur Dokumentation bzw. zur Simulation von Schaltungen / ASICs ins Leben gerufen
- (HW-) Synthese-Werkzeuge erst später
- Zerlegung einer Hardwarebeschreibung in
  - Schnittstellendefinition (Entity)
  - Beschreibung (Architecture): Strukturell, Verhalten oder Mischung

VHDL – Hauptbestandteile

Karlsruher Institut für Technologie

Entity

- Definition der Schnittstelle

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

Architecture

- **Funktionale Verhaltensbeschreibung**
- **Strukturelle Beschreibung** eines Moduls

Configuration

Zuweisung einer Architecture zur verwendeten Entity

VHDL – Entity

Entity

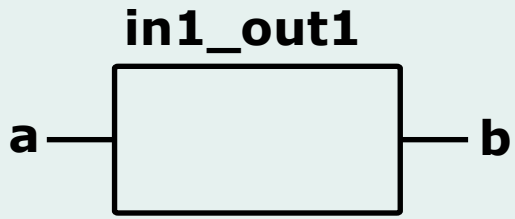
- Definition der Schnittstelle

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

- Festlegung der Datentypen der Ein- und Ausgänge

Entity

```
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
- Zuweisung durch <=

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

- Zuweisung am Blockende

Achtung: Keine mehrfachen Zuweisungen!

Variablen

- Werte innerhalb eines Prozesses
- Zuweisung durch :=

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

- Wertzuweisung sofort

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

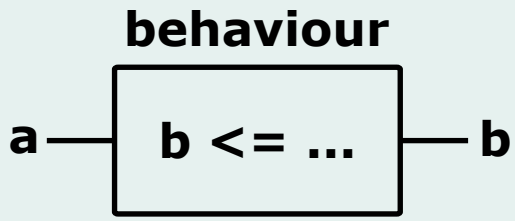
Kontrollfluss

- Generell parallele Abarbeitung
- Sequenzialisierung zumeist **explizit**
- Kontrollfluss:
  - `if ... then begin ... end;`
  - ```
    case signalname is
      when case A => ...
      when others => ...
    end case;
    ```
  - `a <= b when c='1' else '1';`
    ```

VHDL – Architecture

Architecture

- „Füllung“ einer Entity mit Inhalt
- Mehrere Architekturen können für eine Entiy definiert werden



Funktionale Verhaltensbeschreibung (Behaviour)

- Beschreibung der Funktionalität einer Schaltung
- Die interne Struktur wird abstrahiert
- Verhaltensbeschreibung vergleichbar mit einem Programm
- In einfachen Fällen durch eine direkte Funktion
- Funktionales Verhalten durch **nebenläufige Anweisungen**
- Hauptbestandteil einer Verhaltensbeschreibung ist der Prozess

VHDL – Prozess

Karlsruher Institut für Technologie

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

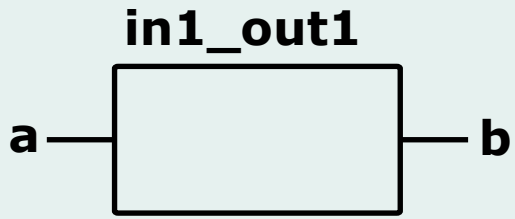
Beispiel

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

VHDL – Beispiel

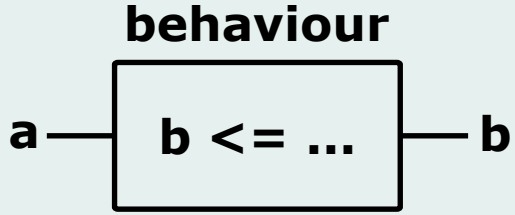
Entity

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



Architecture – Verhaltensbeschreibung

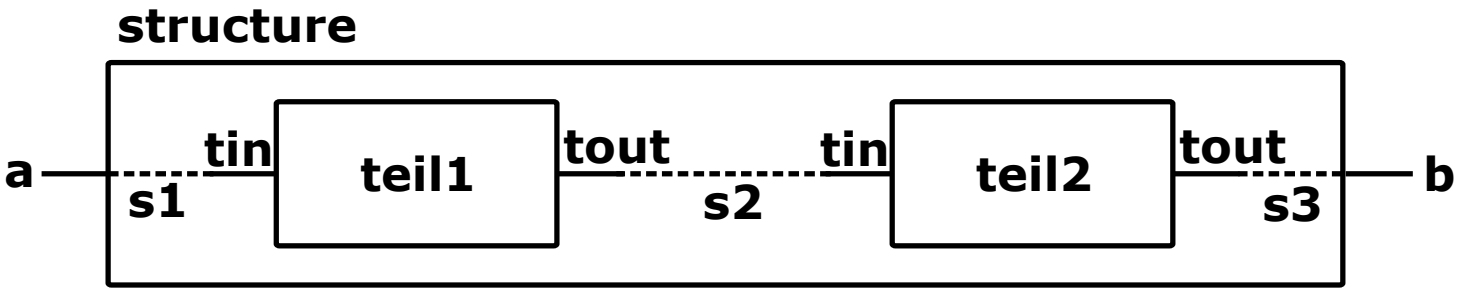
```
architecture behaviour of in1_out1 is
begin
  invertieren : process(a)
  begin
    if a = '1' then
      b <= '0';
    else
      b <= '1';
    end if;
  end process invertieren;
end behaviour;
```



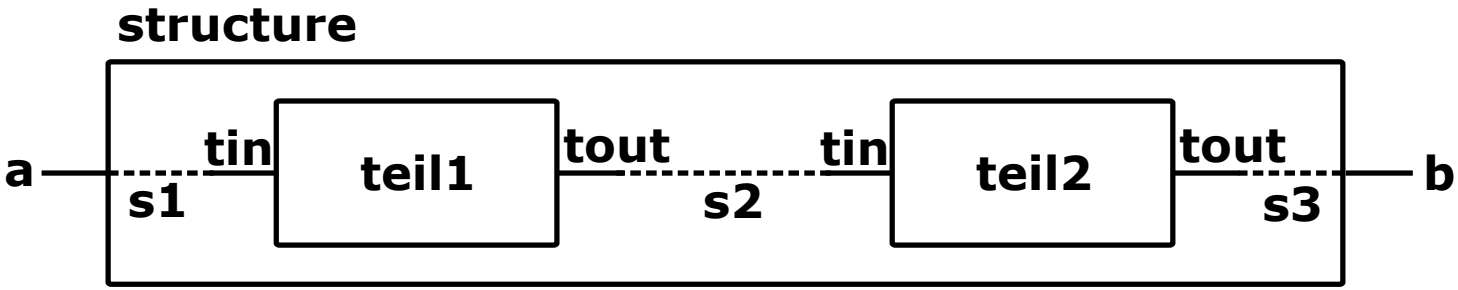
VHDL – Architecture

Strukturelle Beschreibung (Structure)

- Beschreibung einer Schaltung durch Aufteilung in verschiedene Untermodule
- Jedes Untermodul ist die Instanz einer Entity
  $\Rightarrow$ rekursiv $\Rightarrow$ Verhaltensbeschreibung
- Verbindung erfolgt durch Signale
- Deklaration der Schnittstelle der Untermodule (Component)
- Instanziierung der Untermodule



VHDL – Beispiel



Architecture – Strukturelle Beschreibung

```
architecture structure of in1_out1 is
  component teil1
    port (tin : in bit; tout : out bit);
  end component;
  component teil2
    port (tin : in bit; tout : out bit);
  end component;
  signal s1, s2, s3 : bit;
begin
  t1 : teil1 port map (s1, s2);
  t2 : teil2 port map (tout => s3, tin => s2);
  s1 <= a;
  b <= s3;
end structure;
```

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

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

Aufgabe a) Signale und boolesche Funktionen

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

- Bibliotheksaufruf
- Beschreibung der Funktion
- boolescher Beschreibung
- Wertetabelle

Welche weiteren Alternativen finden sich?

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

Einladen benötigter Bibliotheken

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

Lösung: Bibliotheksaufruf

```
c <= a XOR b;
```

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

Lösung: 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;
```

Lösung: boolesche Beschreibung

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

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

Lösung: Wertetabelle

```
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';
```

Lösung: Alternative

```
if (a='0') then
  c <= b;
else
  c <= not b;
end if;
```

Aufgabe b) 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.

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

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

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

Lösung: Schnittstellenbeschreibung

```
entity Counter is
 port (
  clk, n_rst : 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;
```

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

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

Verhaltensbeschreibung

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

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); -- 64 Zustände
begin

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

end architecture;
```

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

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, data_in)`
- Zustand häufig über Zustandsautomaten gehandhabt
- Verwendung logischer und arithmetischer Operatoren
- if/case/when zur Kontroll- und Datenflusssteuerung

Karlsruher Institut für Technologie

Lösung: Verhaltensbeschreibung

Prozess zerfällt in asynchrones Rücksetzen und eigentlichen Zählvorgang.

```
p_counter:
process (n_rst, clk,
         direction,
         enable)
begin
 -- asynchronous reset
 if n_rst='0' then
  count <= 0;

 -- 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: 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.

- Erweitern Sie die Verhaltensbeschreibung aus der vorherigen Teilaufgabe um eine Lösung, bei der das Überlaufssignal außerhalb des Prozesses erzeugt wird.

Lösung: 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 niedrigstwertige 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);
  signal store : std_logic; -- Zustandsspeicher
begin
```

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

```
p_counter: process(n_rst, clk, direction, enable)
begin
  -- asynchrones Rücksetzen
  if n_rst='0' then
    count <= 0; -- unsigned
    store <= '0';

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

    -- Bit 0 des Zählers merken
    store <= count(0);

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

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

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

      end if;

    end if;
  end process;
```

Schaltungsentwurf mit VHDL

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 architecture;
```

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

Aufgabe c) VHDL-Entwurfsprozeß I

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

Karlsruher Institut für Technologie

Aufgabe c) VHDL-Entwurfsprozeß I

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

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

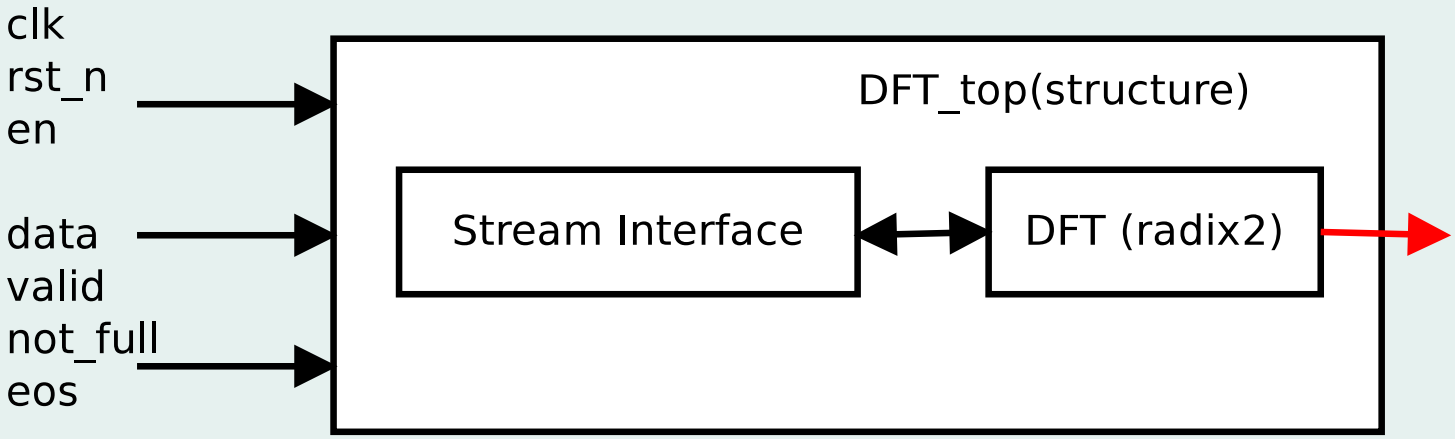
    -- Eingangs-Stream
    data      : in  std_logic_vector(C_DATA_SIZE-1 downto 0);
    valid     : in  std_logic;
    not_full  : out std_logic;
    eos       : in  std_logic
  );
end entity;
```

VHDL-Entwurfsprozess I

Strukturverfeinerung:

Realisierung einer spezifizierten Funktion durch Verschaltung von Komponenten mit einfacherer Funktionalität

Lösung: Architektur



(roter Pfeil: sinnvollerweise Ausgabe der transformierten Daten)

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

```
architecture structure of DFT_top is
  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;
```

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

```
stream_instance : stream_interface
 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
port map (
   clk       => clk;
   rst       => rst;
   data      => data_s2f;
   valid     => valid_s2f;
   not_full  => not_full_f2s;
   eos       => eos_s2f
);

end structure;
```

Karlsruher Institut für Technologie

VHDL-Entwurfsprozess

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
  ```

Schaltungsentwurf mit VHDL

Karlsruher Institut für Technologie

Configuration

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

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;
```

Karlsruher Institut für Technologie

VHDL-Entwurfsprozess II - Zähler

In einer VHDL-Beschreibung sei ein Prozess wie folgt beschrieben. Hierbei sei `count` vom Typ `unsigned(7 downto 0)`:

```
process(clk,count)
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;
```

Karlsruher Institut für Technologie

VHDL-Entwurfsprozess II - Zähler

- 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.

VHDL-Entwurfsprozess II - Zähler

- 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.

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

Karlsruher Institut für Technologie

VHDL-Entwurfsprozess II - Zähler

- 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.
  - Was fehlt in der Schaltungsbeschreibung, um auch in der Simulation eine korrekte Funktion zu gewährleisten?

Karlsruher Institut für Technologie

VHDL-Entwurfsprozess II - Zähler

- 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.
  - Was fehlt in der Schaltungsbeschreibung, um auch in der Simulation eine korrekte Funktion zu gewährleisten?

  $\Rightarrow$ Initialisierung (z.B. mittels Rücksetzsignal)

Karlsruher Institut für Technologie

VHDL-Entwurfsprozess II - Zähler

- 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 die korrekte Funktion des Zählers auch in der Simulation gewährleistet ist.

Karlsruher Institut für Technologie

VHDL-Entwurfsprozess II - Zähler

- 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 die korrekte Funktion des Zählers auch in der Simulation gewährleistet ist.

⇒ `process(clk,count)` → `process(clk,count,rst)`

```
if clk'event and clk='1' then →
if rst='1' then count<="00000000"
elsif
clk'event ...
```

Karlsruher Institut für Technologie

VHDL-Entwurfsprozess II - Zähler

- 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?

Karlsruher Institut für Technologie

VHDL-Entwurfsprozess II - Zähler

- 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.

Karlsruher Institut für Technologie

VHDL-Entwurfsprozess II - Zähler

- 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?

Karlsruher Institut für Technologie

VHDL-Entwurfsprozess II - Zähler

- 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?

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

Ausblick

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

Kontakt

Karlsruher Institut für Technologie

Übungsleiter

- Martin Schindewolf
  `schindewolf@kit.edu`
  0721/608-46048
  Raum 314.1
- Oliver Mattes
  `mattes@kit.edu`
  0721/608-48066
  Raum 308.2

Webseite

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

Karlsruher Institut für Technologie

Zentralübung Rechnerstrukturen im SS 2012

Fragen des Rechnerentwurfs: Fertigung und Hardwareentwurf

David Kramer, Prof. Dr. Wolfgang Karl

Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung

26. April 2012