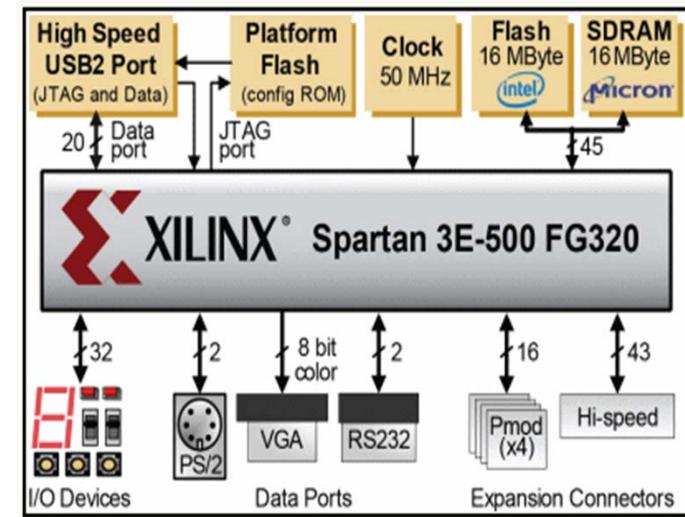


Computer Engineering

CE TI4 SS2011



HTM – SHF - SWR

CE TI4 SS12

Vorlesung: Raum **Di: 1265 Mi: 265**

Zeit: Di : 8 :¹⁵ – 9 :⁴⁵ & 10 :⁰⁰ – 11 :³⁰ ; cw:12-13 Raum 1265

Zeit: Mi : 8 :¹⁵ – 9 :⁴⁵ & 10 :⁰⁰ – 11 :³⁰ ; cw:12-26 Raum 265

Folien: **CETI4_no01_vw01_cw12_120321_vXX.ppt**

Präsentation: Michael Schäfers Raum: 780

email: **CE.S12S@Hamburg-UAS.eu**  : 040 42875-**8155**

home:

<http://users.informatik.haw-hamburg.de/~schafers/>

semi priv  : 040 22815594 (VoIP)

data: https://pub.informatik.haw-hamburg.de/home/pub/prof/schafers/S12S_CE

Labor: Raum 709

Labor-Betreuung: Silke Behn

Raum: 708

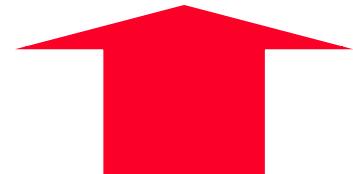
Tutorium: Raum ?

Tutor: Jan Quenzel

(Jan.Quenzel@haw-hamburg.de)

Folien im PUB

- Dokumentation fürs Labor ist unter **emil / Moodle** zu finden
- das "Schäfers"-PUB befindet sich unter:
https://pub.informatik.haw-hamburg.de/home/pub/prof/schafers/S12S_CE
- alle Folien finden sich im PUB nach der Vorlesung



Das müssen Sie sich heute unbedingt notieren!!

- emil/Moodle befindet sich unter:

<http://www.elearning.ls.haw-hamburg.de>



Übersicht

- Grundsätzliches
- Wiederholung & Begriffsabgleich
- RTL & Pipeline & Multi-Cycle-Datapath
- ASM-Charts (allgemein)
- Q-Format
- ASM-Charts (hinführend auf das Labor)
- Synchronisation
- Aufbau Mikroprozessor
- Mikroprozessor-Peripherieeinheiten
- Programmierung Mikrocontroller
- Externe Beschaltung
- Rechnerstrukturen
- ...

Zunächst Sie

- Je nach Antwort(en) kommt mehr oder weniger zu mir
- Wer hatte nicht bei mir DT?
- Wer hatte nicht bei mir Programmieren?
- Wer hatte nicht je "irgendwas" bei mir?

Achtung!

- **DT, PR1, PR2** und GS werden vorausgesetzt.
- Hier wird viel **VHDL & C** gemacht.
- VHDL war Thema in DT und wird hier nur noch vereinzelt abgerundet.
Es wird auf bereits bekannte Programmier-Kenntnisse zurückgegriffen.

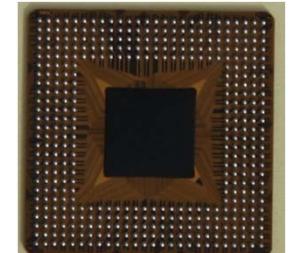
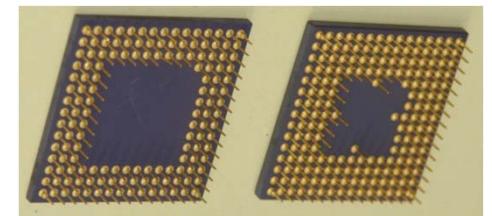
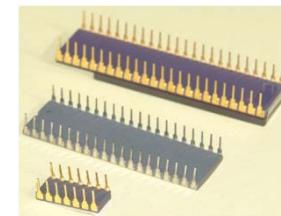
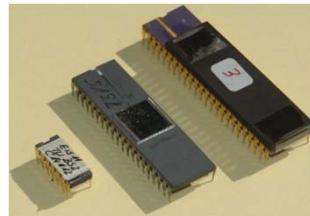
Wer steht gerade da vorn?

Michael Schäfers

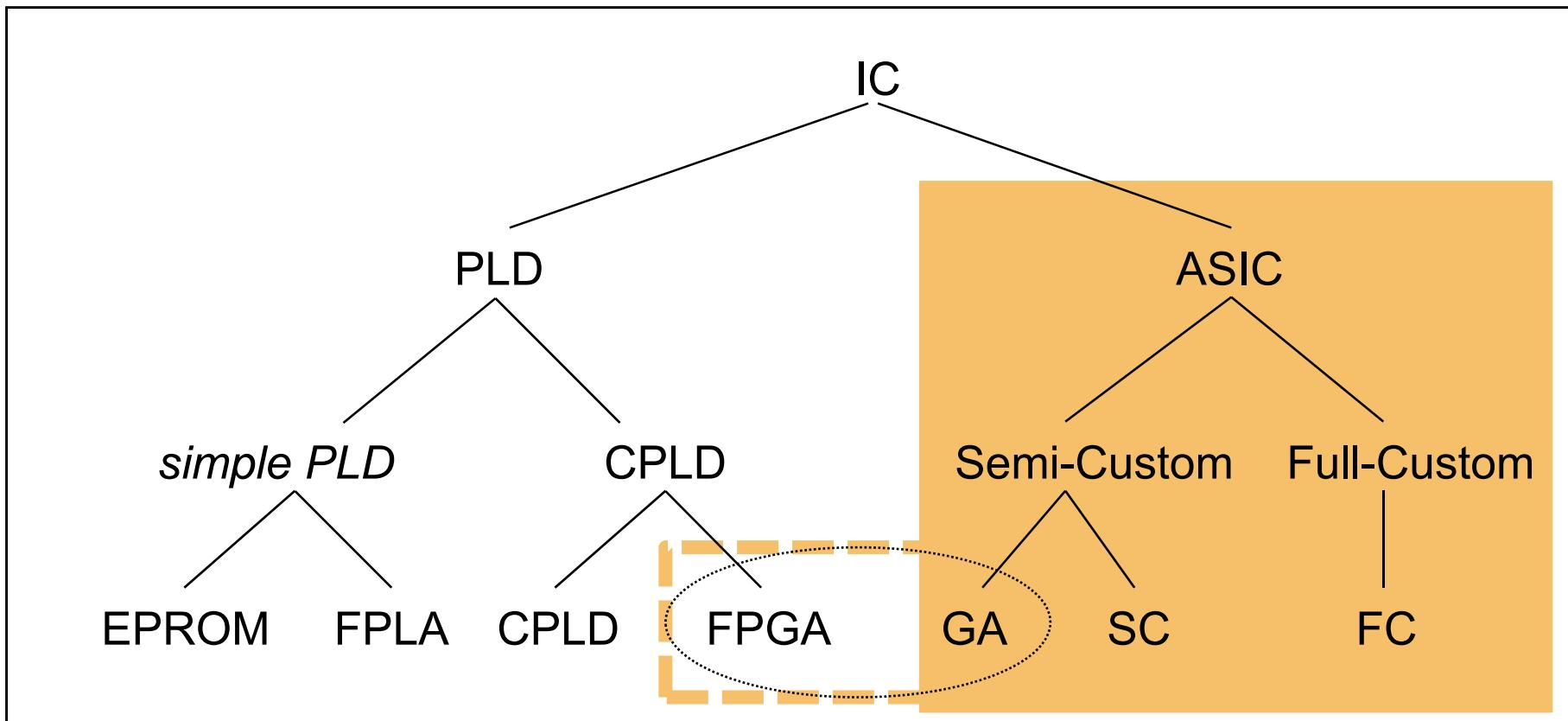
Informatikstudium
+
Promotion
an der TU Braunschweig

Berufstätigkeit (3.1995-3.2002)
Nokia Telecommunications
→ Nokia Networks
→ NSN (*Nokia Siemens Networks*)
Düsseldorf
R&D → **ASIC Entwicklung**

HAW Hamburg seit April 2002



ASICs und andere ICs



CPLD **C**omplex **P**rogrammable **L**ogic **D**evice

EPROM **E**lectrically **P**rogrammable **R**ead **O**nly **M**emory

FPGA **F**ield **P**rogrammable **G**ate **A**rray

FPLA **F**ield **P**rogrammable **L**ogic **A**rray

PLD **P**rogrammable **L**ogic **D**evice

ASIC **A**pplication **S**pecific **I**C

FC **F**ull **C**ustom **I**C

GA **G**ate **A**rray

IC **I**ntegrated **C**ircuit

SC **S**tandard **C**ell

ASICs und andere ICs (2)

- IC (**I**ntegrated **C**ircuit): Integrierte Schaltkreise - meist in CMOS gefertigt.
- PLD (**P**rogrammable **L**ogic **D**evice) - Produktion "abgeschlossen" - Kunde programmiert "fertiges" IC.
- ASIC (**A**pplication **S**pecific **I**C) - Kunde hat Einfluss auf Produktion (HW-Beschaffenheit) des IC.
- Full-Custom - Kunde hat Einfluss auf Masken aller Layer - Produktion entsprechend der Kundenwünsche - Kunde hat Kontrolle über jeden Transistor.
- Semi-Custom - Kunde hat begrenzten Einfluss auf Masken der Layer.
- SC (**S**tandard **C**ell) zwar werden letztlich Masken aller Layer durch Kundenwünsche bestimmt, aber Kunde darf nur vom Hersteller vorentwickelte Basiszellen verwenden - Kunde kann also nur "gestellte" Basis-Zellen Transistoren verwenden.
- GA (**G**ate **A**rray) - Produktion der unteren Masken, Layer bereits abgeschlossen - Teile des IC sind bereits vorgefertigt. Die oberen Verdrahtungslayer werden entsprechend der Kundenwünsche vorgefertigt.
- Weiterhin unterscheidet man zwischen
 - "Channeled" : Verdrahtung nur in reservierten Kanälen ("alte Technik" aus der Zeit "der wenigen Layer") und
 - "Channelless" : Verdrahtung (unter Beachtung der "Design Rules") überall möglich.

Bemerkung

Die Abkürzungen:

- IC **I**ntegrated **C**ircuit
- CPLD **C**omplex **P**rogrammable **L**ogic **D**evice
- FPGA **F**ield **P**rogrammable **G**ate **A**rray

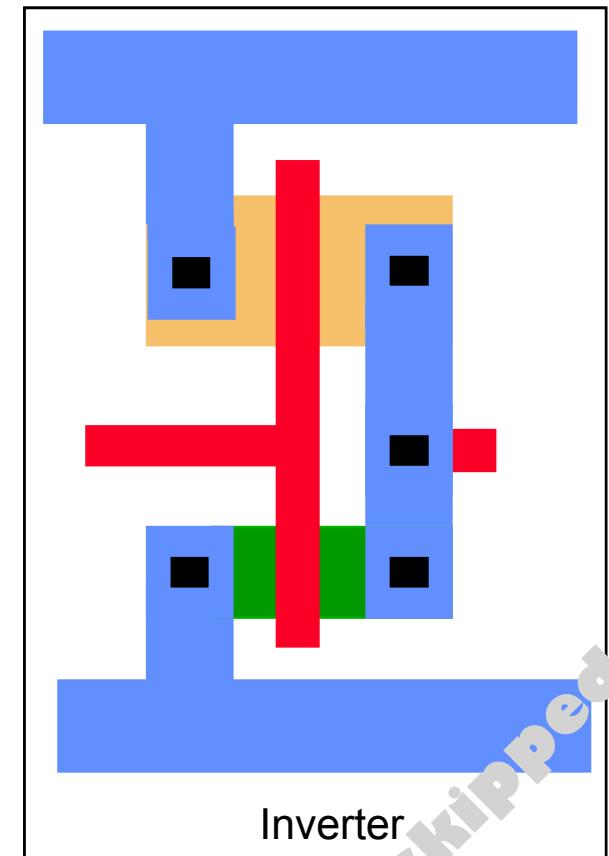
werden Sie häufiger hören.

Und bei mir:

- ASIC **A**pplication **S**pecific **I**ntegrated **C**ircuit

Full-Custom *(geometrische Layout-Ebene)*

- **maßstabsgetreue** Vergrößerung der schichtenweisen Strukturen im fertigen Chip
- geometrische Entwurfsregeln (Mindestabstände und Mindestbreiten)
- simuliertes Layout ist die Schnittstelle des Full-Custom-Designers
- geometrische Layout wird in textuelle Beschreibungsform (GDS2, CIF2.0) gewandelt. Nach dem Tape-Out werden hiermit die Masken für die Chipfertigung erstellt
- mittels Extraktor Überführung in Schaltkreisebene
- Zeitmodell: kontinuierliche Realzeit
- beobachtbare Werte: kontinuierlicher Wertebereich für Spannungen, Ströme
- Arbeitsmittel: SPICE



Konsequenz

- GA+SC+FC sind "schneller" und höher integriert als FPGA
- Je mehr vorgefertigt ist, (also je weniger Einfluss der spezifische Kunde hat)
 - desto höher die Fertigungsstückzahlen beim Halbleiterhersteller
(Halbleiterhersteller kann Produkt auch anderen Kunden für andere Applikationen anbieten)
- Je mehr entsprechend der Kundenwünsche gefertigt wird
 - desto mehr erfüllt das IC die Kundenwünsche
(bzw. desto weniger enthält IC "Unnötiges")
- Je nach Stückzahlen, die Kunde (Entwickler, der IC in seinem Produkt verwendet) vom Halbleiterhersteller abnimmt, desto mehr lohnt eine Produktion entsprechend der Kundenwünsche
- Also: $\mu\text{P}/\mu\text{C} \leftrightarrow \text{FPGA} \leftrightarrow \text{GA} \leftrightarrow \text{SC} \leftrightarrow \text{FC}$
- Soweit die Idee
und dann kommt das Marketing sowie die Wunderwelt der Marktwirtschaft
- Hier - im Labor : in DT: CPLD und in CE: FPGA

Skipped

Sprechstunde

- vor und nach Vorlesung
 - CE-Labor
 - sofern Zeit (P2-Labor am Fr, Raum 1101b)
 - per Absprache
-
- meine Email-Adresse: **CE.S12S@hamburg-uas.eu**
 - Kontaktdaten auf jeder Vorlesungs-Start-Folie
-
- Feedback
 - Kommentare, Korrekturen, Verbesserungsvorschläge, Anregungen jeglicher Art, ... sind willkommen

Termine

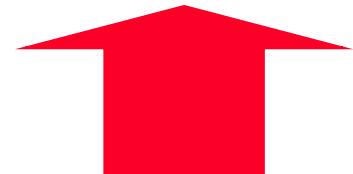
- „üblich“ / „meist“ der letzte Termin als Frage&Antwort-Stunde zur Klausurvorbereitung
- 16 Zeit-Slots 2xMi + 14xDo
Do. 17.Mai (cw20) ist keine Vorlesung \Rightarrow zunächst **16 Slots**
- $16 \times 4 = 64 \approx 16 \times 4$ \Rightarrow ca. **16 Termine**
 \Rightarrow *wir gehen (fast) „immer“ über die volle Zeit*
- vermutlich (16 x 4) leider incl. (1 x „F&A“) *wann F&A ?*
- Achtung der "DT2"-Teil findet *(in der Vorlesung)* in den ersten 2-3 Wochen statt !!!
Bestimmt aber über lange Zeit das Labor.

Hinweise

- Kein Script ☹ - „nur“ die Vorlesungsfolien im PUB
- Eigenverantwortliches Mitschreiben und Nacharbeiten
Nicht alles wird angeschrieben oder verteilt ☹☹☹
Insbesondere dieses Mal viel an der Tafel und (nur?) gesagt
- Selbststudium (üben, Manuals, Bücher, ...)
- **Achtung Zeit ist knapp!**
- **Unbedingt auf das Labor/Praktikum vorbereiten!**
- Sprache der Informatik ist englisch
- In den Email-Verteiler eintragen

Wdh.: Folien im PUB

- Dokumentation fürs Labor ist unter **emil / Moodle** zu finden
- das "Schäfers"-PUB befindet sich unter:
https://pub.informatik.haw-hamburg.de/home/pub/prof/schafers/S12S_CE
- alle Folien finden sich im PUB nach der Vorlesung



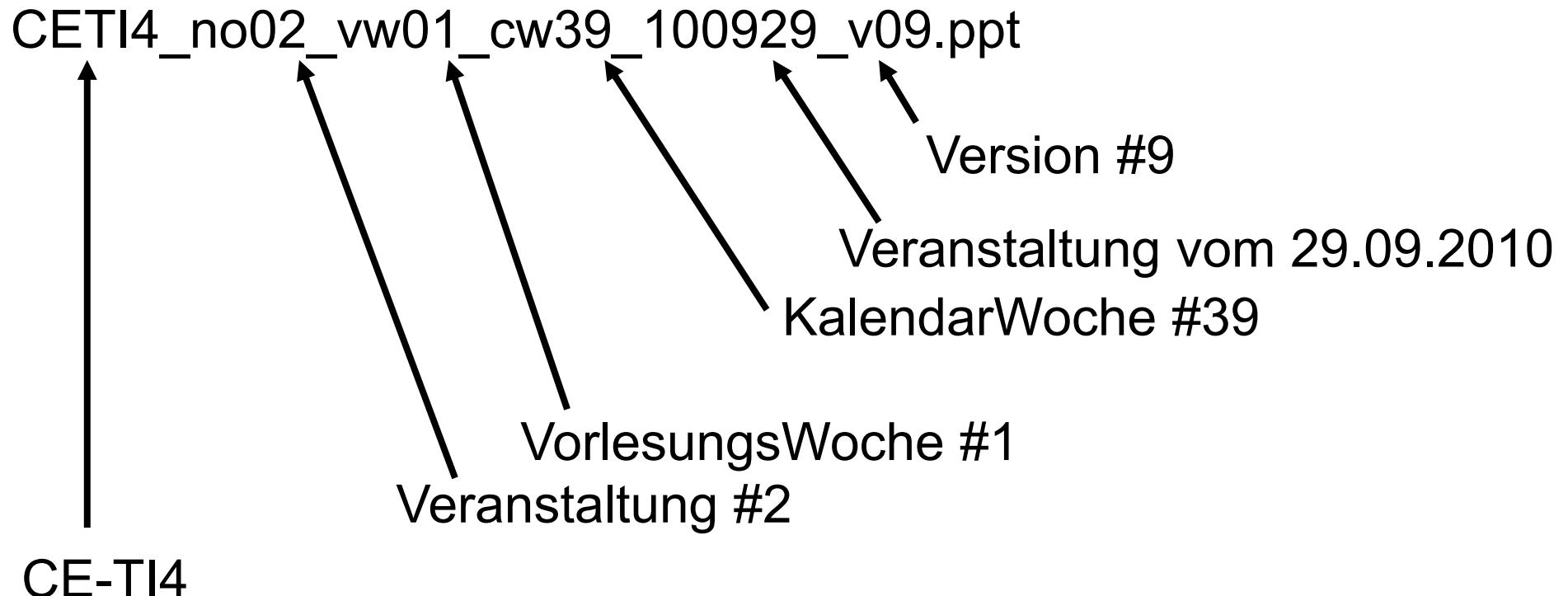
Das müssen Sie sich heute unbedingt notieren!!

- emil/Moodle befindet sich unter:

<http://www.elearning.ls.haw-hamburg.de>



System der Dateinamen im PUB



Aktuelle Folien liegen unter:

https://pub.informatik.haw-hamburg.de/home/pub/prof/schafers/S12S_CE

Der letzte Zyklus liegt unter:

https://pub.informatik.haw-hamburg.de/home/pub/prof/schafers/S11S_CE

Hintergrund zu CE_{PO2008} (allgemein)

- Mit der PO2008 wurden die "alten" Vorlesungen DT2, RS und CE_{früher} zu CE zusammengefasst
- CE bzw. CE_{NEU} bzw. CE_{PO.2008} lief im SS2010 zum ersten Mal
- 4+2 Veranstaltung ⇒ 4LVS und 8 Labortermine
- 8 CP
- Themen:
 - FPGA
 - μController
 - Rechnerstrukturen
 - Kommunikation von SW und HW

Hintergrund zu CE_{PO2008} (DT-Anteil)

- Mit der PO2008 wurden die "alten" Vorlesungen DT2, RS und CE_{früher} zu CE zusammengefasst
- DT2_{früher} baute direkt auf DT1_{früher} auf und RS_{früher} direkt auf DT2_{früher}
- DT2_{früher} heißt nun intern auch DS (Digitale Systeme)
- CE baut direkt auf DT auf - die DT-Kenntnisse sind weiterhin sehr wichtig

DT1_{früher} und DT2_{früher}

- wurden in Vergangenheit von Herrn Thomas Canzler und Bernd Schwarz gehalten
- basierten auf einer ausgearbeiteten Vorlesung von Jürgen Reichardt und Bernd Schwarz
- DT1_{früher} und DT2_{früher} waren gut in abgedeckt in
VHDL-Synthese - Entwurf digitaler Schaltungen und Systeme; Reichardt, Schwarz; Oldenbourg
- Neu von Herrn Reichardt ist das Lehrbuch Digitaltechnik; Reichardt; Oldenbourg
- Mehr in der Literaturliste

Literatur (1)

- [1] J. Wakerly: Digital Design Principles & Practices. Prentice Hall 2006, 4th edition
- [2] S. Brown; Z. Vranesic: Fundamentals of Digital Logic with VHDL Design. McGraw Hill 2005, 2nd edition
- [3] P. J. Ashenden: Digital Design. An Embedded Systems Approach Using VHDL. Morgan Kaufmann 2008
- [4] D. D. Gajski: Principles of Digital Design; Prentice Hall 1997
- [5] J. Reichardt; B. Schwarz: VHDL Synthese. Entwurf digitaler Schaltungen und Systeme; Oldenbourg 2009, 5. Auflage
Digilent NEXYS2 Board
- [6] W. Wolf: Computers as Components: Principles of Embedded Computing System Design.
Morgan Kaufmann 2008, 2nd edition
HAW SECURITYv2 Board
- [7] W. Wolf: High-Performance Embedded Computing; Morgan Kaufmann 2007
- [8] A. Sloss, D. Symes, Ch. Wright: ARM System Developer's Guide; Morgan Kaufmann 2004
- [9] U. Brinkschulte, Th. Ungerer: Mikrocontroller und Mikroprozessoren; 2. Auflage Springer 2007
- [10] D. A. Patterson, J. L. Hennessy: Computer Organisation & Design; The Hardware/Software Interface; Morgan Kaufmann 2008, 4th edition

"Literatur" (2)

- Digilent Nexys2 Board Reference Manual (Moodle)
- Anleitung Xilinx / ModelSim
- The VHDL Designers Guide ($\geq 3.$ ed)
→ IEEE Standard for VHDL RTL Synthesis (ch21, p633-665)
Peter Ashenden
Morgan Kaufmann, ≥ 2008
- XST (Xilinx® Synthesis Technology) User Guide
Xilinx Online-Dokumentation
- A VHDL Primer ($\geq 3.$ ed)
J.Bhasker
Prentice Hall, ≥ 1998
- A Guide To VHDL ($\geq 2.$ ed)
S.Mazor, P.Langstraat
KAP/Kluwer, ≥ 1993
- Reuse Methodology Manual – For System-On-A-Chip Designs ($\geq 3.$ ed)
M.Keating, P.Bricaud – Synopsys, Mentor
KAP/Kluwer, ≥ 2007

"Literatur"/Unterlagen

emil/Moodle

- Punkt 2 Praktikum Allgemein
unbedingt gründlich lesen



Zunächst:

- **Diligent Nexys2 Board Reference Manual**
- Spartan 3E FPGA DataSheet
- Anleitung Xilinx / ModelSim

The screenshot shows a Moodle course page with the following structure:

- Header:** www.elearning.haw-hamburg.de/course/view.php?id=661
- Breadcrumbs:** schlag nach > HAW > BIB > KnowHow > TV > Stuff
- Course Title:** Computer Engineering SoSe 2012
- Navigation:** Startseite > Kurse > Lehrveranstaltungen 2012 > Technik und Informatik > CE_TI_12
- Aktuelle Termine:** Es gibt keine weiteren Termine. (Zum Kalender... Neuer Termin...)
- Neue Nachrichten:** (Keine Nachrichten im Forum)
- Navigation:** Startseite, Meine Startseite, Website, Mein Profil, Kurse (Lehrveranstaltungen 2012: Design, Medien und Information, Life Sciences, Technik und Informatik: CE_TI_12, Wirtschaft & Soziales, Information und Organisation 2012), Einstellungen (Kurs-Administration: Abmelden aus CE_TI_12, Mein Profil).
- Themen dieses Kurses:** Foren (Allgemeine Nachrichten und Ankündigungen), Vorlesung, Praktikum Allgemein (FPGA: Diligent Nexys2 Board Reference Manual, Spartan 3E FPGA DataSheet, Anleitung Xilinx / ModelSim, Vorbereitung der FPGA Labor Aufgaben, Arm TI Board: LPC2468 UserManual, Einführung HiTop, 64 MB Atmel SPI Serial Flash Memory AT25DF641 Datasheet, Schaltplan TI Board, Terminal).
- Aufgabe1: Shift Register**

Labor

- Sind Sie bereits im Labor angemeldet ?
- Falls nein, sofort! zu Frau Behn
- Es gibt Teams ⇒ TeamNr holen (z.B. "S1T2")
- Vermutlich 1 Aufgabenzettel/Termin (wir werden sehen)
- Jedenfalls gilt:
 - 1. Aufgabenzettel muss am 1.Termin abgenommen werden
 - 2. Aufgabenzettel muss am 2.Termin abgenommen werden
 - 3. Aufgabenzettel muss am 3.Termin abgenommen werden
 - Danach sehen wir weiter

- Bei Rückstand gilt Anwesenheitspflicht im Tutorium(!)
- Wer Rückstand hat und **nicht** im Tutorium erscheint wird des Labors verwiesen bzw.
hat die CE-PVL des SS2012 verwirkt

⇒ **CE Tutorium Termin**

- Am Ende des Semesters müssen alle Aufgaben gelöst sein.
- Gibt es diesbezüglich Fragen?

Vorbereitung, Abgabe & emil / Moodle

- Ist irgendwem emil/Moodle unbekannt?
- emil / Moodle ist wichtige Datenbasis für CE. Dort liegen **wichtige Unterlagen**.
- Schicken Sie Ihre Vorbereitungsunterlagen als Email an **CE.S12S@Hamburg-UAS.eu** Spätestens bis So. 23:59Uhr **vor** Antritt des Labors.
- Schicken Sie Ihre Lösungen als Email an **CE.S12S@Hamburg-UAS.eu**.
- Nachdem alle(!) Aufgaben gelöst sind, laden Sie Ihre Lösung gekennzeichnet mit Team-Nr bei emil/Moodle hoch.
- Vorlesungsfolien und Aufgaben finden Sie im PUB.
- Sofern noch nicht mit emil / Moodle vertraut → nachholen!
→ www.elearning.ls.haw-hamburg.de
- Password/Anmeldekennung für emil/Moodle "Computer Engineering" (SoSe 2012) ist:
"cep"

Labor

- Email an `CE.S12S@Hamburg-UAS.eu`
- Eine Kopie der Email wird automatisch an Frau Behn weitergeleitet
- Die Email hat ein Subject (Betreffzeile), das wie folgt aufgebaut ist
`<TeamNr>_<Vorlesung>_<Aufgabe>_<Code-Version>`
- Alle Buchstaben sind groß und Trennung mit Underscores ("_)
- Vorbereitung als Version 0 (`v0`)
- Abgabe als `v1` (bzw. `v2, v3, ..., v9` für nachfolgende Versionen - hoffentlich nicht nötig)
- Konkret z.B. Aufgabe1:

<code>SxTy_CEP_TI4_A1_v0</code>	Vorbereitung
<code>SxTy_CEP_TI4_A1_v1</code>	Abgabe
- **x** und **y** sind geeignet zu ersetzen ☺
- Wir erinnern uns?
Programmieren - Formales - Regeln zur Code-Abgabe ☺

Labor

Email

- Subject: **SxTy_CEP_TI4_A#_V#**
- In der Email **nur!** Source-Code & Dokumentation
 - .vhd (VHDL-Code),
 - .do (Scripte - sofern vorhanden, wave.do)
 - .tcl (Scripte - sofern vorhanden)
 - .ucf (Pinning-Datei),
 - .pdf (Text/Dokumentation)
 - .jpg (Bilder - Skizzen gern handgezeichnet & abgetragen)
 - .c (C-Code), PDF-Dateien, JPGs
- In der Email die Dateien **nicht!** komprimieren
- Unterschiedliche Dateien (einer Aufgabe) haben unterschiedliche Namen

Labor

emil / Moodle:

- Unter emil/Moodle darf (dann später) auch Zusätzliches abgelegt werden
- Dort gern auch komprimiert
- Dort unbedingt ebenfalls Team-Nr als Überschrift angeben
Am besten Subject wie bei Email verwenden
- Spätestens am Ende des Semester muss/müssen die Abgabe(n) auch unter email/Moodle liegen - sonst **keine** PVL.

WEB-Page
für Probleme
in der
7.Etage (!)

http://www.informatik.haw-hamburg.de/fehler_melden.html

Fehler melden - Mozilla Firefox

File Edit View History Bookmarks Tools Help

schlag nach HAW Java TV

FEHLER melden

Fakultät Technik und Informatik

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

HAW Hamburg

Design, Medien und Information Life Sciences Technik und Informatik Wirtschaft und Soziales Suche Go

DEPARTMENTS > Department Informatik > Labore > TI-Labor > Fehler melden

Fehlermeldung im "Labor fuer Technische Informatik"

Felder mit * sind Pflichtfelder!

Wo tritt der Fehler auf?

* Labor Bezeichnung

07.01 (BSP, SEP1+2, PLP)
07.09 (GSP, GTP, CEP)
07.60 (Master, WPP, POP)
07.65 (RNP, VSP, BSP, WPP, POP)
Sonstige Räumlichkeiten

* Fehlerbeschreibung

Ihre Daten:

* Titel/Vorname
* Nachname
* E-Mail
* Zugriffscode

Sicherheitseingabe

t3esa

Done

Labore

Labor 07.01

Labor 07.09

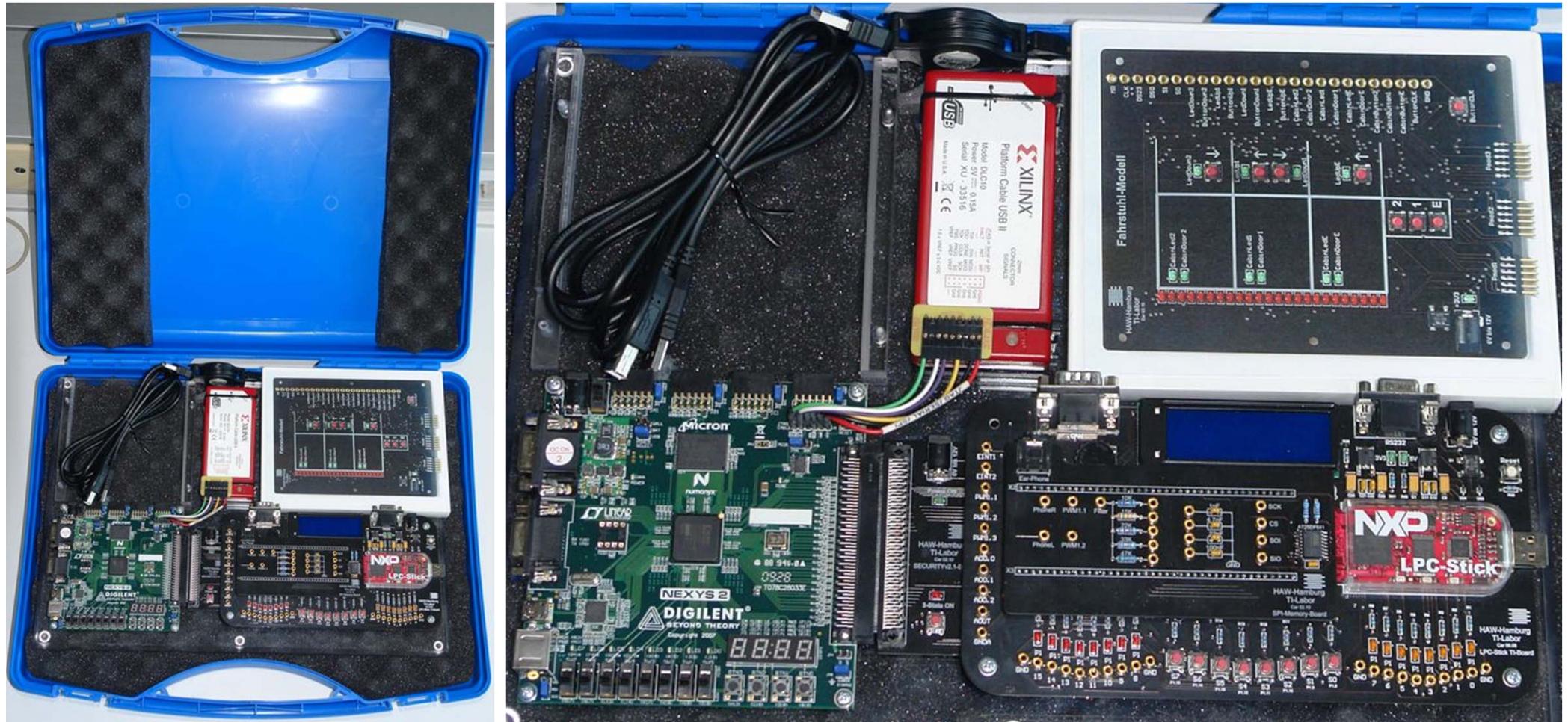
Labor 07.60

Labor 07.65

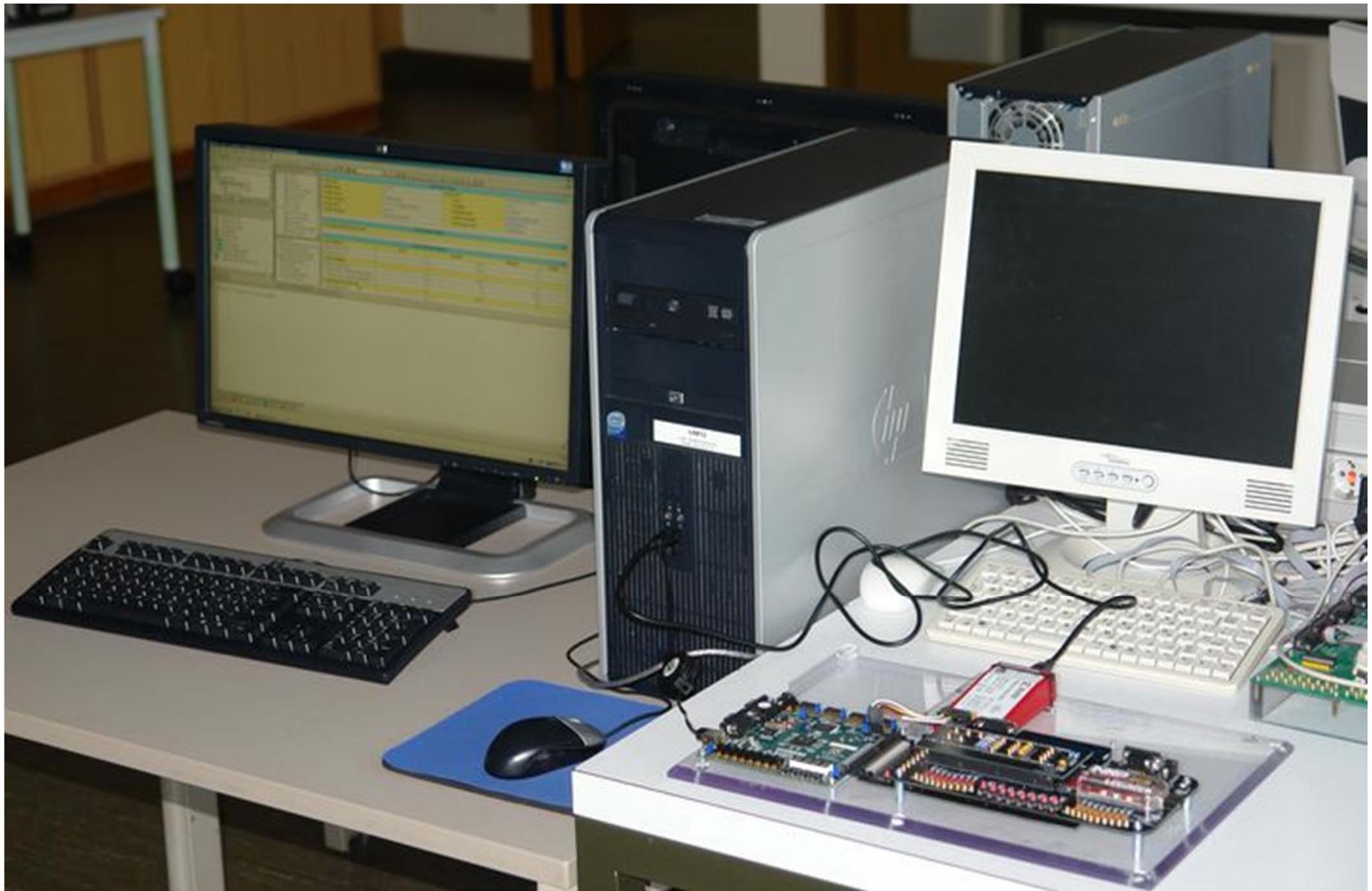
Labor : Wo sind die CE-Koffer?



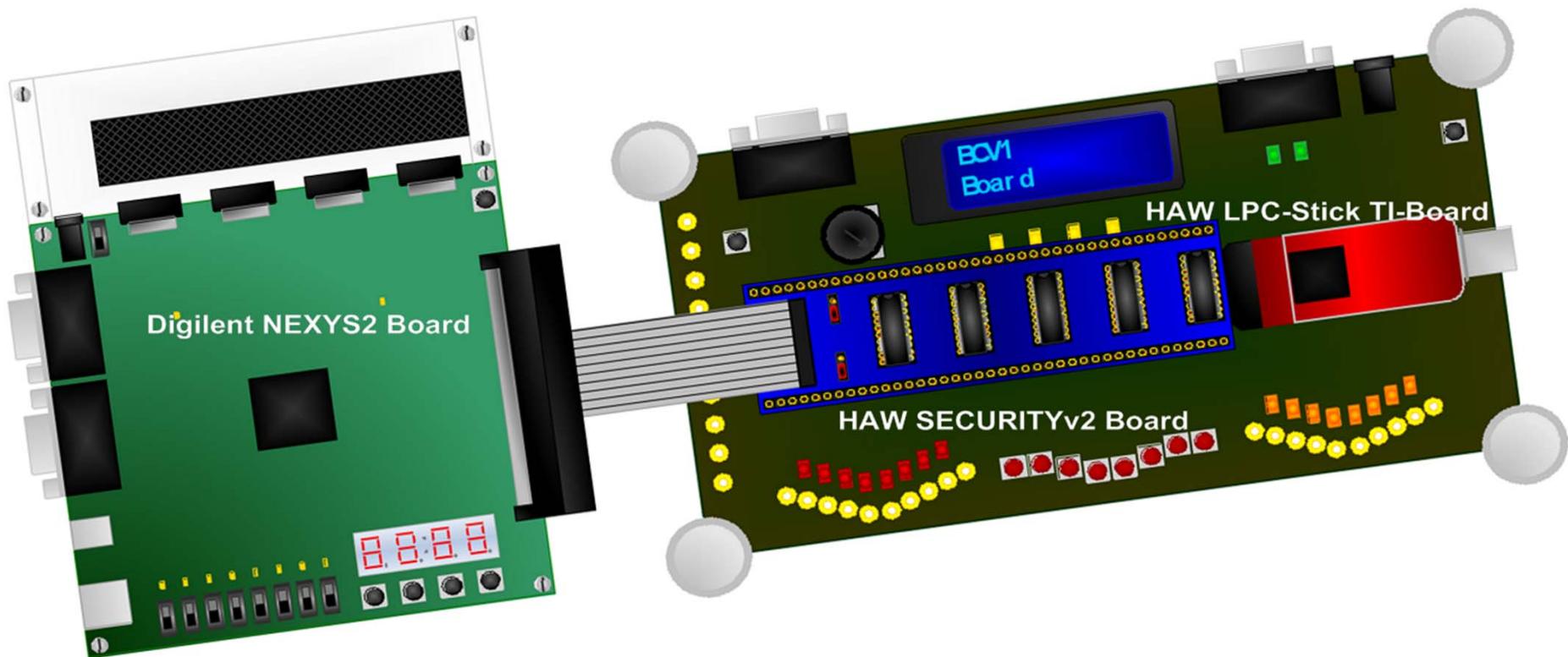
Labor : Der CE-Koffer



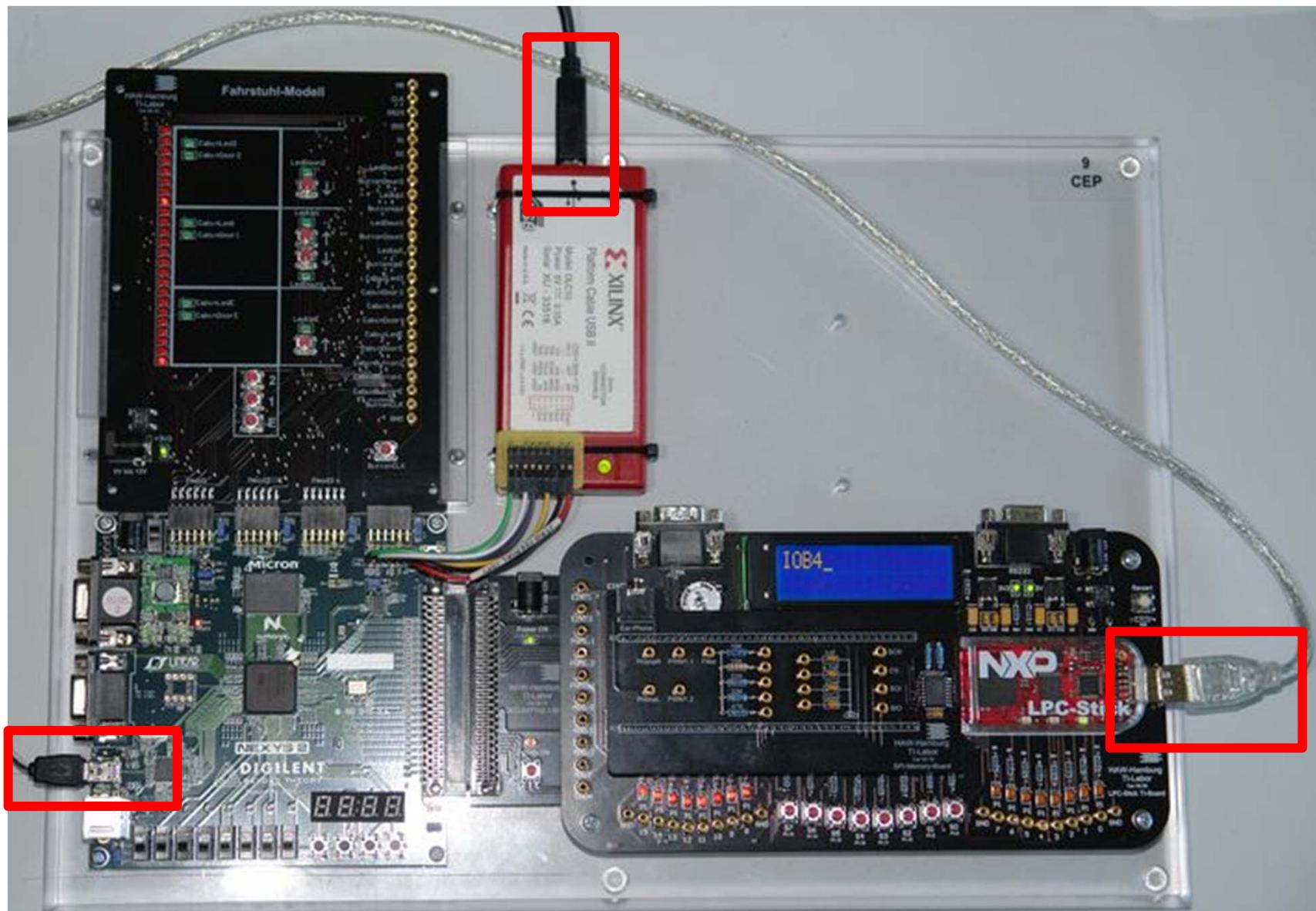
Labor : Arbeitsplatz



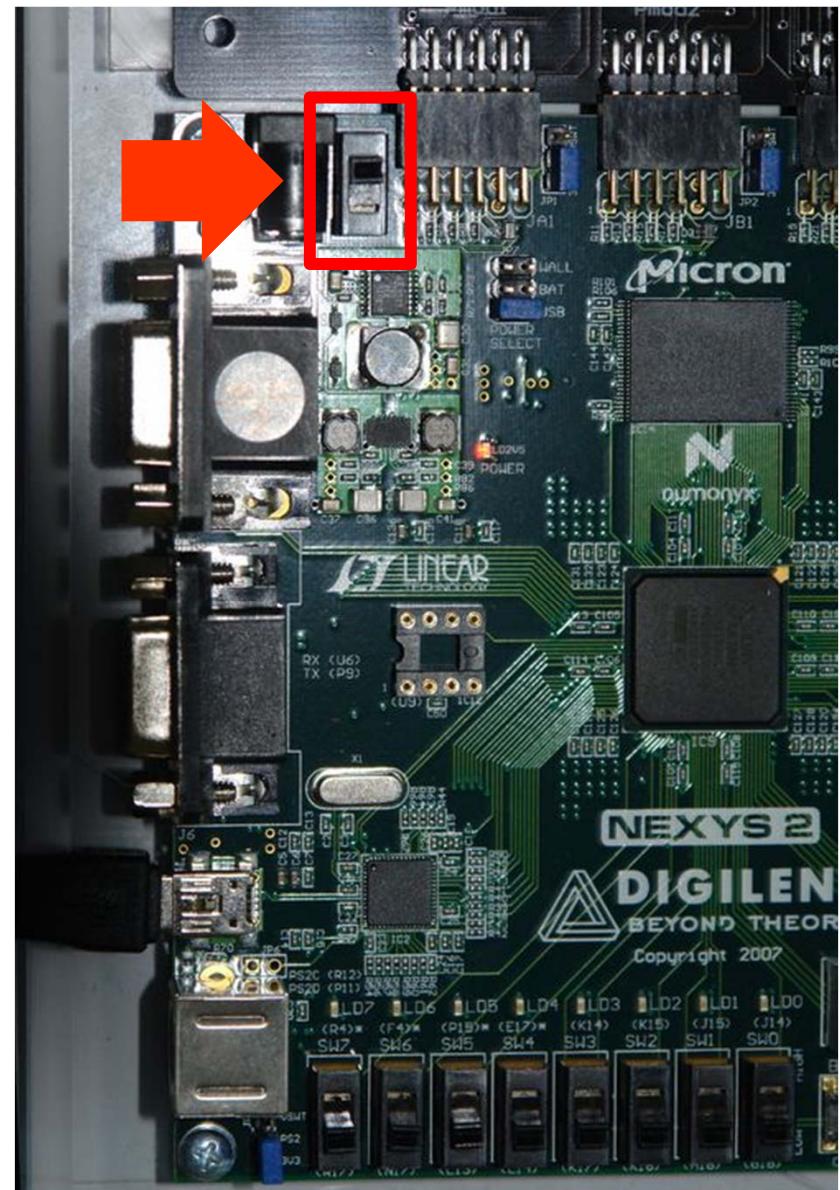
Labor



Labor



Labor

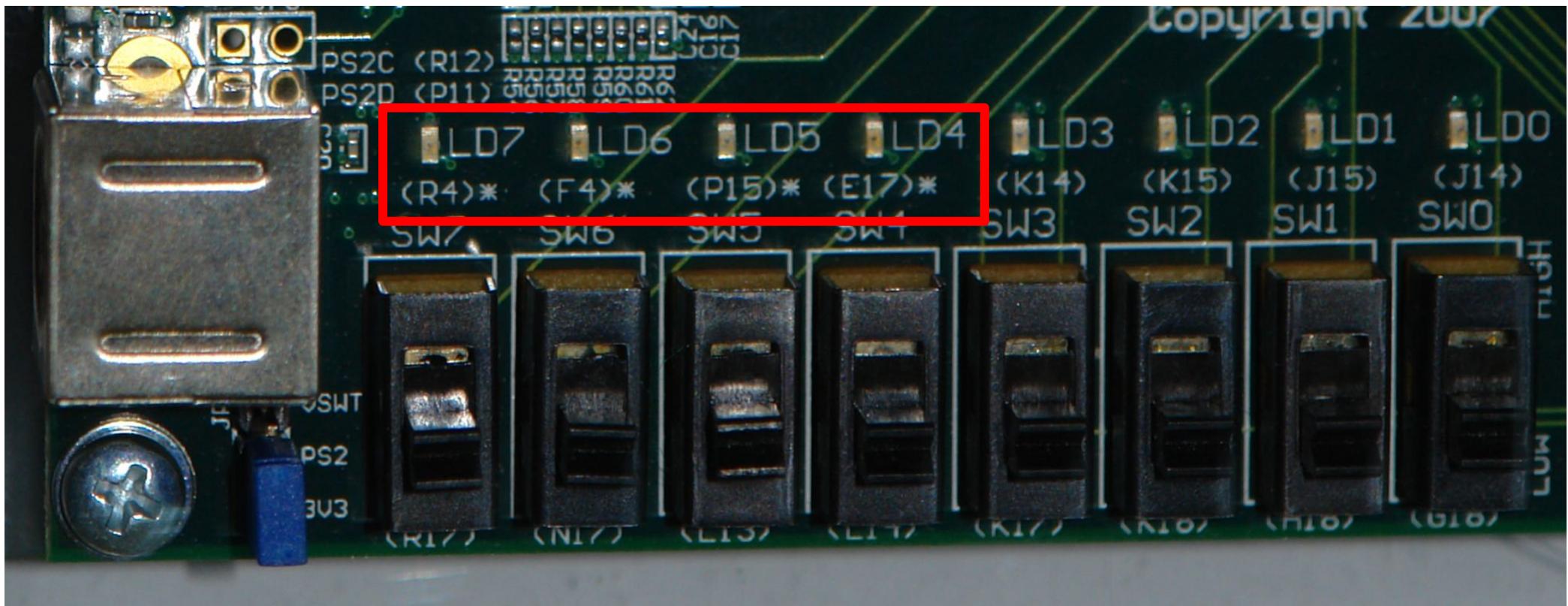


Labor : Digilent Nexys2 Board

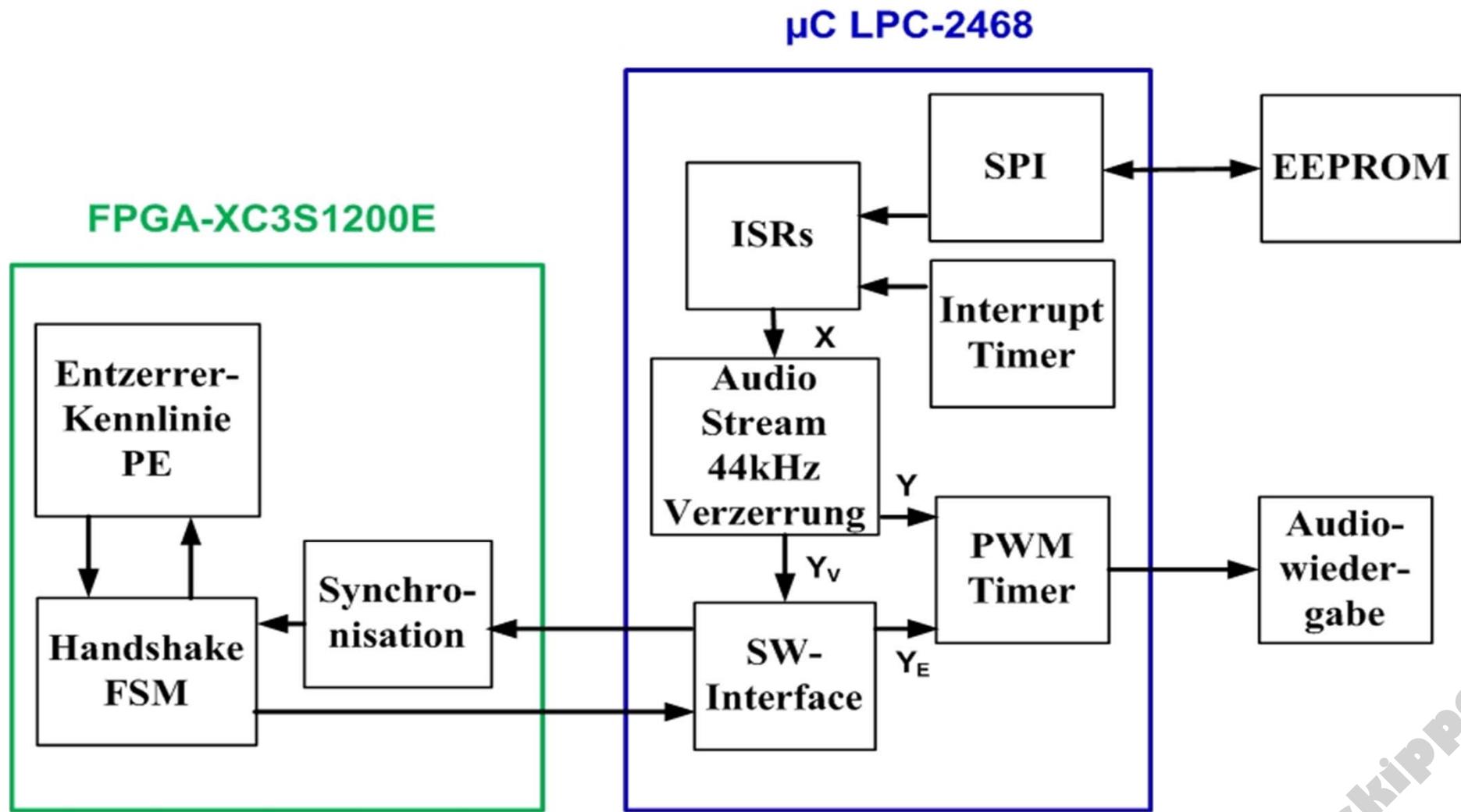
- Achtung!

PINs zu den LEDs (siehe auch Digilent Nexsys2 Reference Manual)

LD7→R4/**P4**; LD6→F4/**E4**; LD5→P15/**P16**; LD4→E17/**E16**

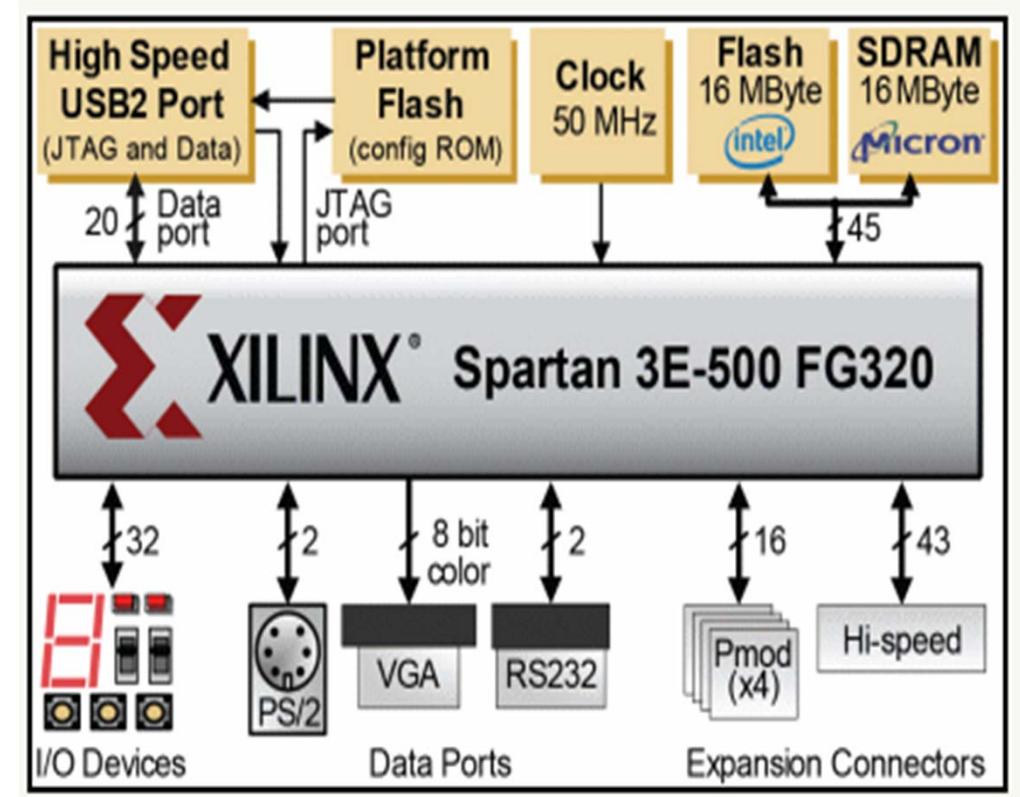


CE-Labor : System / "Rahmen für Aufgaben"



Digitent Nexys2

- Xilinx XC3S1200E
19500 LEs, 63kB RAM, 28 MUL
- USB2: board power, device config.
and high-speed data transfers
- 16MB fast Micron SDRAM
- 16MB Intel Flash Flash ROM
- Xilinx Platform Flash ROM
- 50MHz Oscillator + Socket
- 59 GPIOs
- On-board I/O includes 8 LEDs,
4 7-segment display, 4 pushbuttons,
8 slide switches
- 119\$=91€
- ADC-, DAC-, Video-Decoder-, RS232-, 10Mbit Ethernet-Module



<BREAK>

Wdh. DT Erinnern wir uns? Kennen wir das (noch)?

Was ist ein Δ -Zyklus ?

Wie testen Sie / wie erzeugen Sie Stimuli ?

Abgrenzung Variable \leftrightarrow Signal ?

Wie gestalten Sie Ihre Waves ?

Abgrenzung Entity \leftrightarrow Architecture \leftrightarrow Component ?

TB/SG/RC ?

Was ist eine Timingsimulation ?

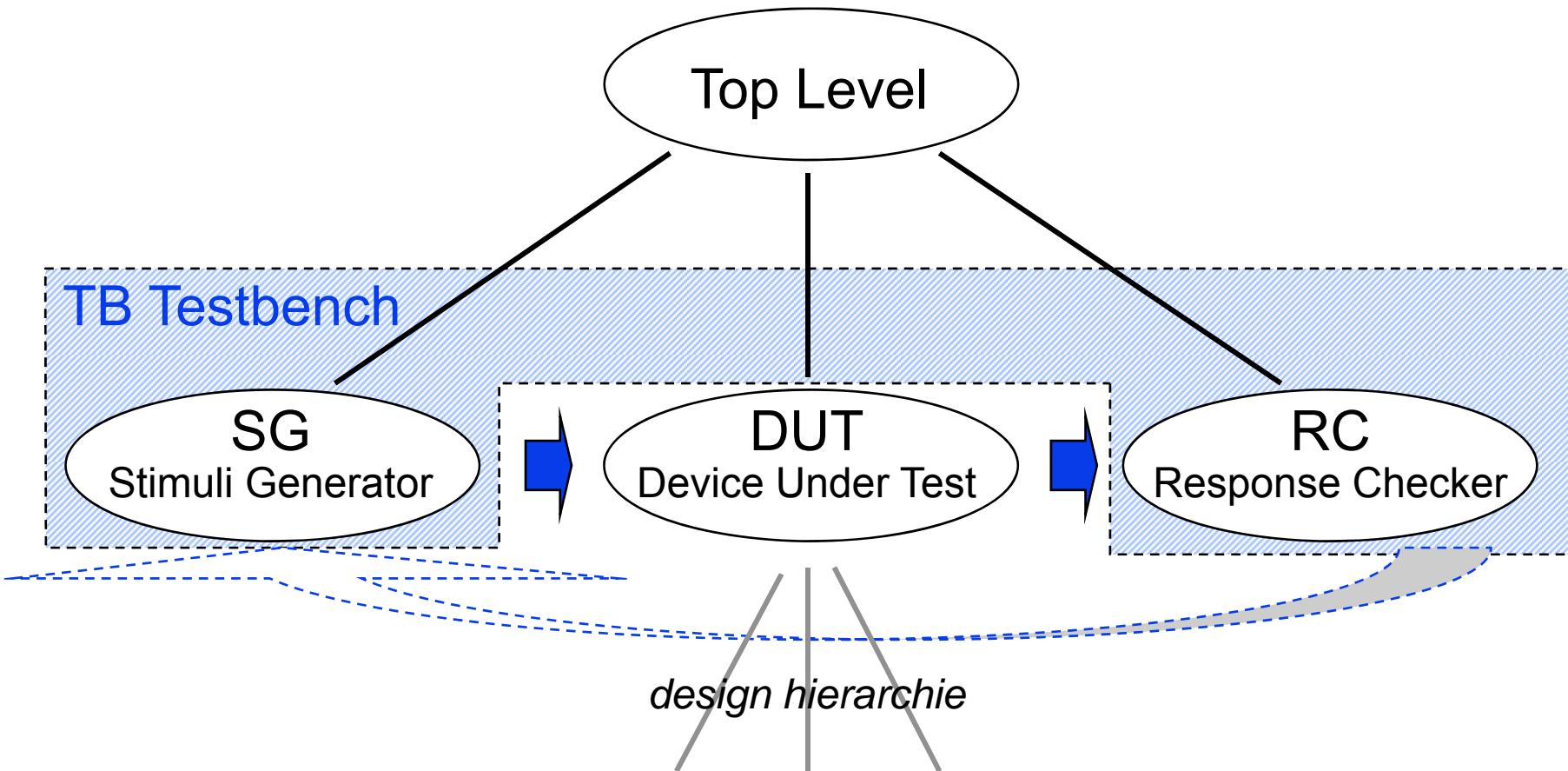
Was ist ein Timing-Loop (=kombinatorische Schleife)?

Was ist ein kritischer Pfad?

Achtung! **Nicht**-"WS11/12-DT"-Teilnehmer

Unbedingt https://pub.informatik.haw-hamburg.de/home/pub/prof/schafers/S11W_DT_an schauen!

Top Level / Testbench



Wir erinnern uns: Assignments

2 Zuweisungen

- $:=$ Variable Assignment

Wert-Zuweisung erfolgt unmittelbar sofort an Variable

Variable Assignments werden innerhalb von Prozessen genutzt

Variablen werden innerhalb von Prozessen genutzt für algorithmische temporäre "Größen"

- $<=$ Signal Assignment

Wert-Zuweisung erfolgt verzögert an Signal

Signale repräsentieren HW-Verbindungen und werden genutzt um "Werte" zwischen den Prozessen/Komponenten auszutauschen

- Bemerkung
In VHDL ist die Zeit zweidimensional

Arbeitsbegriffe

Die gleich folgenden Begriffe R-, V-, H-Zeit

- sind Arbeitsbegriffe
- so also nicht in der Literatur anzufinden

Wer stehende Begriffe dafür findet, der möge Sie mir bitte mitteilen

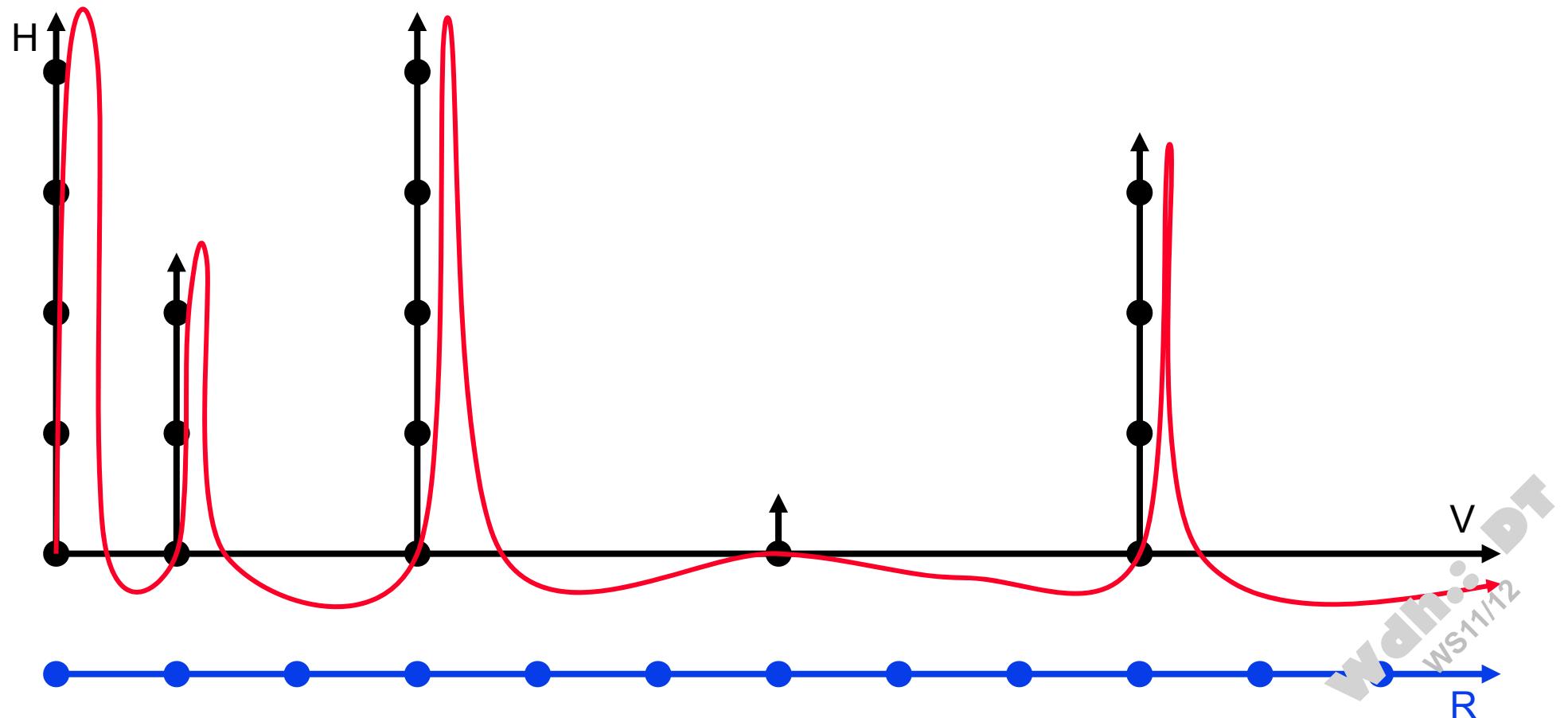
Wdh:: OT
WS11/12

VHDL und die Zeit 1

- reale physikalische Zeit
 - die reale Zeit, wie sie von uns erlebt wird
 - gemessen in Sekunden (konkrete Einheit wird je nach Anwendung gewählt)
- geplante reale physikalische Zeit
 - die reale Zeit, wie sie von uns erlebt werden wird, sofern das Projekt nicht eingestellt oder die Spezifikation modifiziert wird (*andernfalls wird sie nie erlebt ;-)*
 - gemessen in Sekunden (konkrete Einheit wird je nach Anwendung gewählt)
 - kurz **R-Zeit** (für **real**)
- virtuelle physikalische Zeit
 - die direkte Abstraktion der R-Zeit
 - gemessen in Sekunden (konkrete Einheit wird je nach Anwendung gewählt, z.B. ps). Diese Zeit ist diskret - eine eindimensionale Liste von Zeitpunkten.
 - kurz **V-Zeit** (für **virtuell**)
- virtuelle "Hilfs-Zeit"
 - wird zur Modellierung der Parallelität benötigt, diese Zeit läuft während eines Zeitpunkts der V-Zeit
 - gemessen in Delta. Dieses Zeit ist diskret (eine eindimensionale Liste von Zeitpunkten)
 - kurz **H-Zeit** (für **Hilf**)

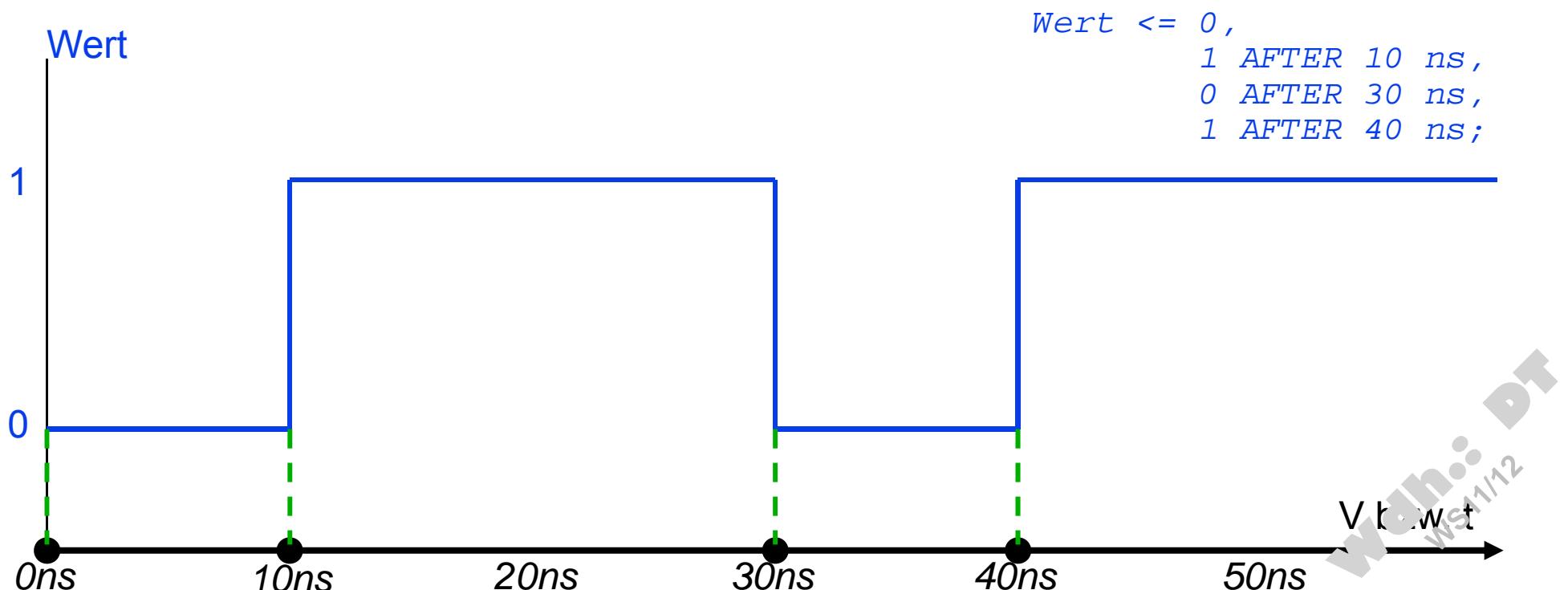
VHDL und die Zeit 2

- zu jedem Zeitpunkt der V-Zeit gibt es genau einen Zeitpunkt der R-Zeit
 injektive Abbildung von V nach R aber nicht surjektiv
- zu jedem Zeitpunkt der H-Zeit gibt genau einen Zeitpunkt der V-Zeit
 die Abbildung von H nach V ist weder injektiv noch surjektiv



VHDL und die Zeit 3

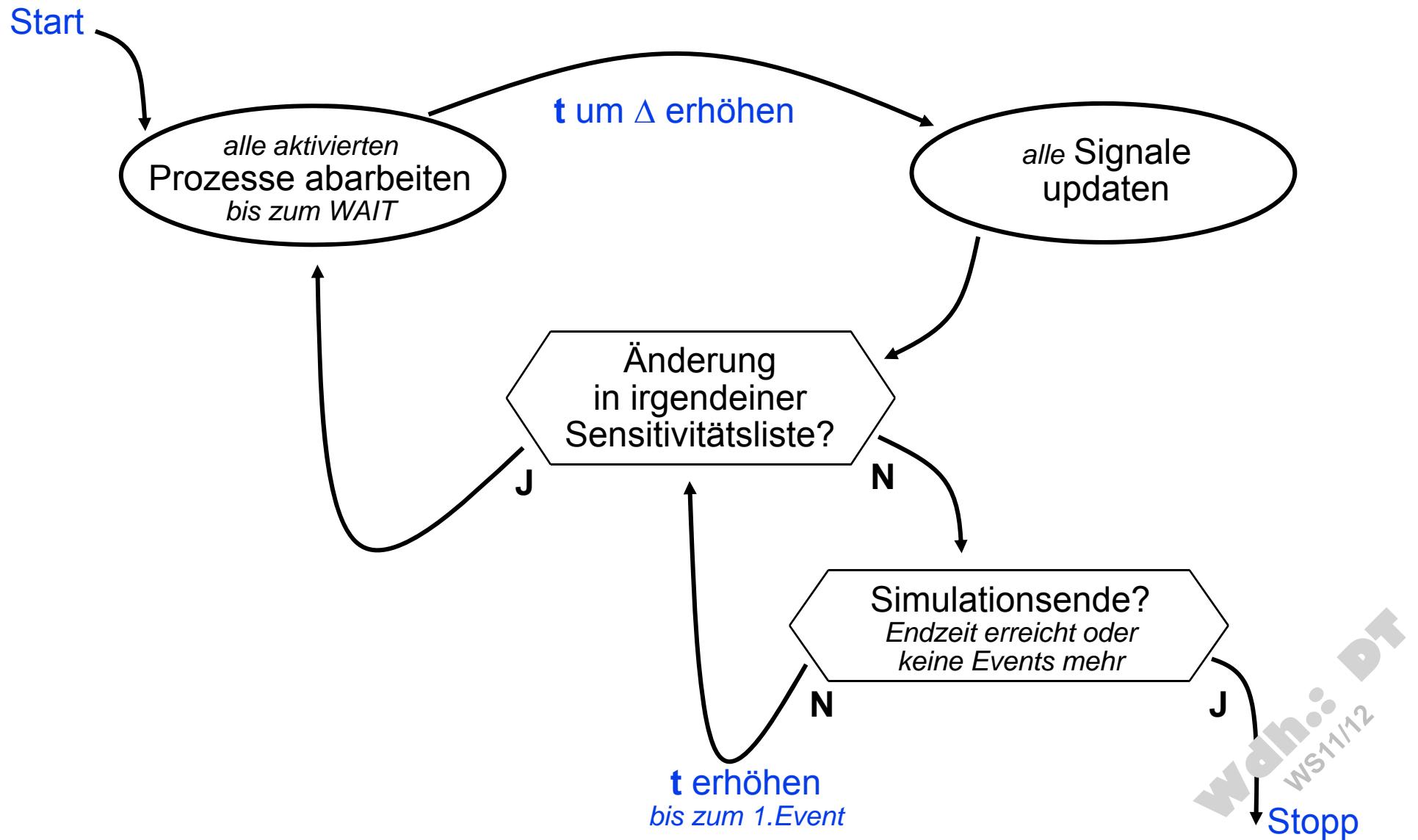
- VHDL (ein VHDL-Simulator) ist Ereignis getrieben (*event driven*)
- jedem Ereignis (Transaktion) kann eine Zeitkoordinatenpaar (v, h) zugeordnet werden, z.B. (4510ps, 8 Δ)
- bei der üblichen graphischen Darstellung mit Waves werden die Deltas unterschlagen, es werden die Werte zum jeweils letzten Delta (H-Zeit) eines Zeitpunkts (V-Zeit) dargestellt



Parallelität und VHDL

- in VHDL passieren "Dinge" entweder
 - sofort in V-&H-Zeit (Variable-Assignment) oder
 - verzögert (Signal-Assignment *und explizites WAIT*)
- zur Realisierung von Parallelität muss auf Signale zurückgegriffen werden
- jede Signalzuweisung löst eine Transaction aus, der ein (V,H)-Zeitkoordinatenpaar zugeordnet wird
Ein Event ist eine Transaction bei der sich was "ändert".
- jede Signalzuweisung "kostet" eine gewisse V/H-Zeit.
 - $x \leq y;$ 0 ns und 1Δ
 - $u \leq v$ AFTER 10 ns; 10 ns

VHDL und die Zeit 4 oder das "VHDL-Scheduling"



Prozess-Verarbeitung in VHDL

- alle Prozesse werden parallel ausgeführt
- Prozesse kommunizieren über Signale
- Prozesse werden (*ohne Unterbrechung*) ausgeführt bis sich selbst suspendieren (auf WAIT bzw. auf die Sensitivitätsliste* laufen). VHDL-Prozesse sind non-preemptive!
- Prozesse haben eine Sensitivitätsliste* (von Eingangssignalen). Wenn eine Veränderung (Event) in der Sensitivitätsliste auftritt wird der zugehörige Prozess (re)aktiviert
- wenn ein Prozess stimuliert wird, reagiert er und wartet danach auf neue Stimuli
- die an ein Signal ausgesendeten Transaktionen werden in einer geordneten Liste gesammelt und heißen **Driver** des Signals. Jeder Transaktion ist ein Bi-Tupel zugeordnet, das den Wert und die Zeit der Transaktion festlegt

* WAITS eingeschlossen

Alles klar ? 😊

- Das VHDL-Simulationsmodell
 - Folie: VHDL und die Zeit 4 oder das "VHDL-Scheduling" ist wichtig für das Verständnis der "Effekte".
- Es erleichtert auf einfache Art die Nachbildung der in der Realität vorhandenen Nebenläufigkeit
- **Das (Simulations-)Ergebnis ist unabhängig vom Scheduling des Simulators!**
Sofern Sie das "System" nicht mit globalen Variablen (Shared Variables) unterlaufen.
- Das VHDL-Simulationsmodell macht die korrekte Modellierung der "Nebenläufigkeit" leicht!

Wdh:: BT
WS11/12

Wir erinnern uns noch einmal 😊

- Variablen und ":=" / Variable Assignment

Wert-Zuweisung erfolgt unmittelbar sofort an Variable

Variable Assignments werden innerhalb von Prozessen genutzt

Variablen werden innerhalb von Prozessen genutzt für algorithmische temporäre "Größen"

Variablen dürfen nur innerhalb von Prozessen verwendet werden.

Variablen haben nicht die Intention HW zu repräsentieren

Variablen unterstützen das sequentielle Denken indem Programmierer stark sind

Variablen lassen sich leichter debuggen

(Debugger sind "primär" für Sequenzen geeignet)

- Signale und "<=" Signal Assignment

Wert-Zuweisung erfolgt verzögert an Signal

Signale repräsentieren HW-Verbindungen

Signale werden genutzt um "Werte" zwischen den Prozessen/Komponenten auszutauschen

Laufzeiten werden "Signalen" modelliert

Codierungsphilosophie (1)

strikte Trennung zwischen kombinatorischer und sequentieller Logik

in der sequentiellen Logik erfolgen nur Signalzuweisungen jeweils zum Reset und der Taktflanke (bzw. Taktpiegel)

in der kombinatorischen Logik (Schaltnetz) wird nur auf Variablen gerechnet

- zunächst werden die Signale auf Variablen umgesetzt
- dann wird auf Variablen gerechnet
- dann werden die Ergebnisse über Signale weitergesendet

Wdh. / Zur Erinnerung: Tipp: Namen

Kennzeichnung im Namen, ob Variable oder Signal (und was für ein Signal)

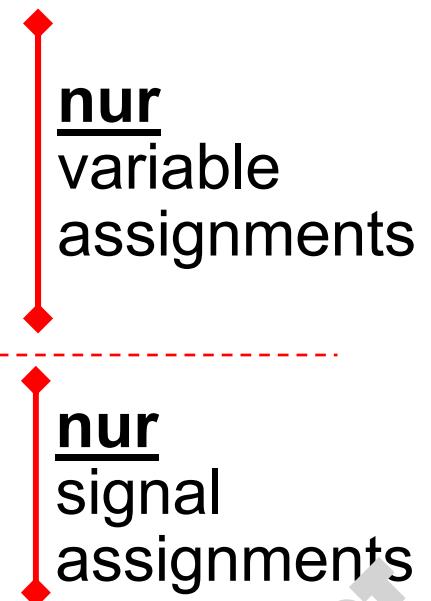
- *name_v* **Variable**
- *name_cs* **Clocked signal**
- *name_ns* **New signal / signal to be clocked**
- *name_s* **Signal (neither „_cs“ nor „_ns“)**

Codierungsphilosophie (2a)

```
exampleIncrementer:  
process ( xi_s ) is  
    variable x_v : integer;  
begin  
  
    -- Schritt 1: Signale abgreifen und auf Variablen umsetzen  
    x_v := xi_s;  
  
    -- Schritt 2: auf Variablen Berechnungen durchführen  
    inc_x_v := x_v + 1;  
  
    -- Schritt 3: Ergebnisse als Signale versenden  
    inc_s <= inc_x_v;  
  
end process exampleIncrementer;
```

Alle Berechnungen finden im 2.Schritt statt.

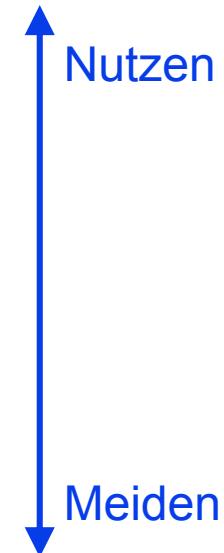
Insbesondere im 3.Schritt gibt es keine Berechnungen, sondern ausschließlich Zuweisungen



Codierungsphilosophie (3)

Kombinatorische Logik

- Bits zusammenstellen (konkatenieren, schieben & "wegwerfen")
- NOT, AND, OR, NAND, NOR
- "=", XOR
- ADD, SUB, "<", ">"
- MUL
- DIV
- Komplizierteres



Bemerkung

- "=" besteht aus Äquivalenz-Gattern
- "<" bzw. ">" besteht aus Subtrahierern

Wdh:: OT
WS11/12

Begriffsklärung

Schaltnetze

- werden gebildet durch **kombinatorische** Logik
- enthalten **keine** Rückkopplungen
- werden in der Praxis auch “kombinatorische Logik” (**combinational logic**) genannt

Schaltwerke

- werden mit kombinatorischer Logik und Rückkopplungen aufgebaut
- durchlaufen über die Zeit **Zustände** (*state*) bzw. eine Sequenz von Zuständen
- die Zustände werden durch **sequentielle** Logik (**sequential logic**) realisiert
- Schaltwerke werden auch als **FSM** (*finite state machine* bzw. endliche Zustands-Automaten) bezeichnet

sequentielle Logik

- ein pegelgesteuertes oder einstufiges Flipflop wird in der Praxis meist als **Latch** bezeichnet
- ein flankengesteuertes, zweistufiges oder Master-Slave-Flipflop wird in der Praxis meist verkürzt als **Flipflop** bezeichnet

Begriffsklärung zur sequentiellen Logik

Latch

- einstufiges Flipflop
- pegelgesteuert (**level sensitive**) i.d.R. vom Takt
- auch: taktzustandsgesteuert aber sogar auch kurz "Flipflop" genannt
- korrekt: einstufiges pegelgesteuertes Flipflop
- hier: Kurzform Latch

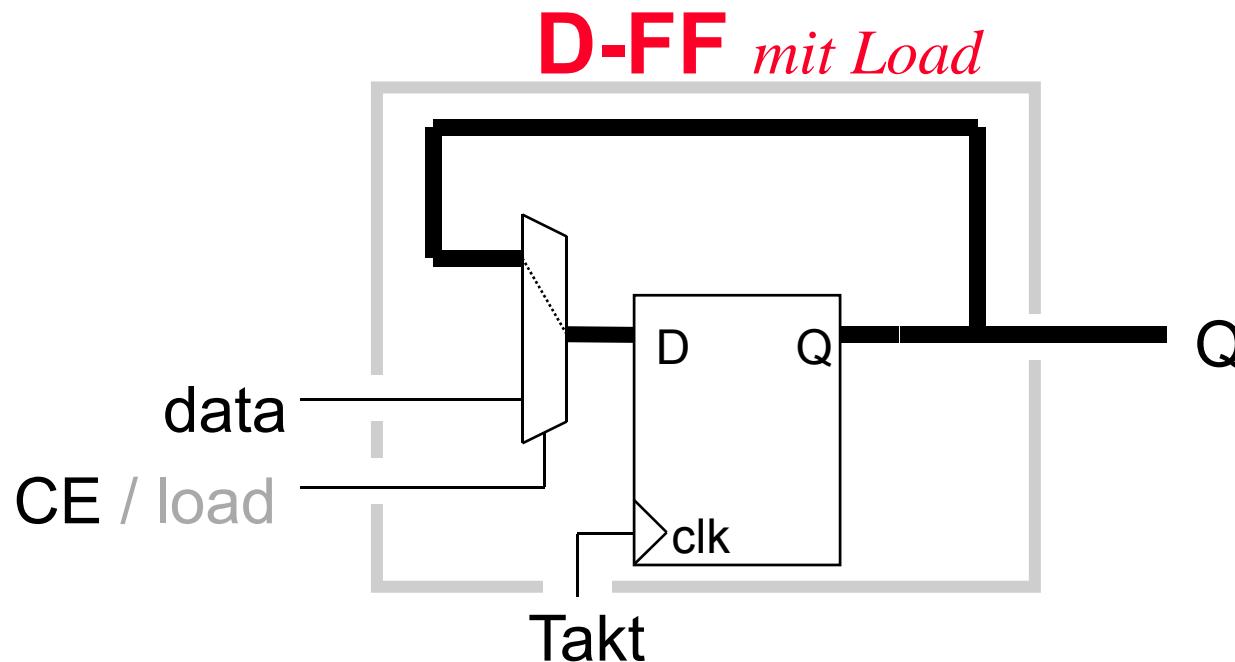
Flipflop

- zweistufig
- flankengesteuert (**edge sensitive**) i.d.R. vom Takt
- auch: taktfleckengesteuert oder aber sogar auch kurz "Latch" genannt
- korrekt: zweistufiges flankengesteuertes Flipflop
- hier: Kurzform Flipflop

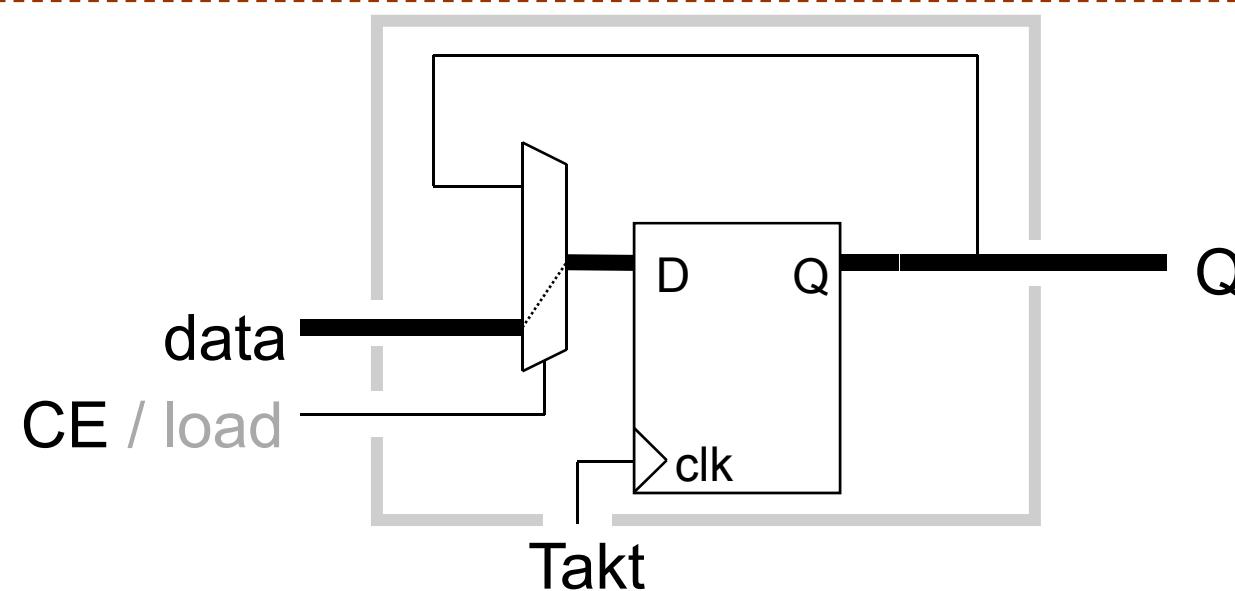
Achtung

- die Namen Latch und Flipflop sind nicht "genormt". Unterschiedlich Bedeutungen sind möglich (s.o.)

Speichern



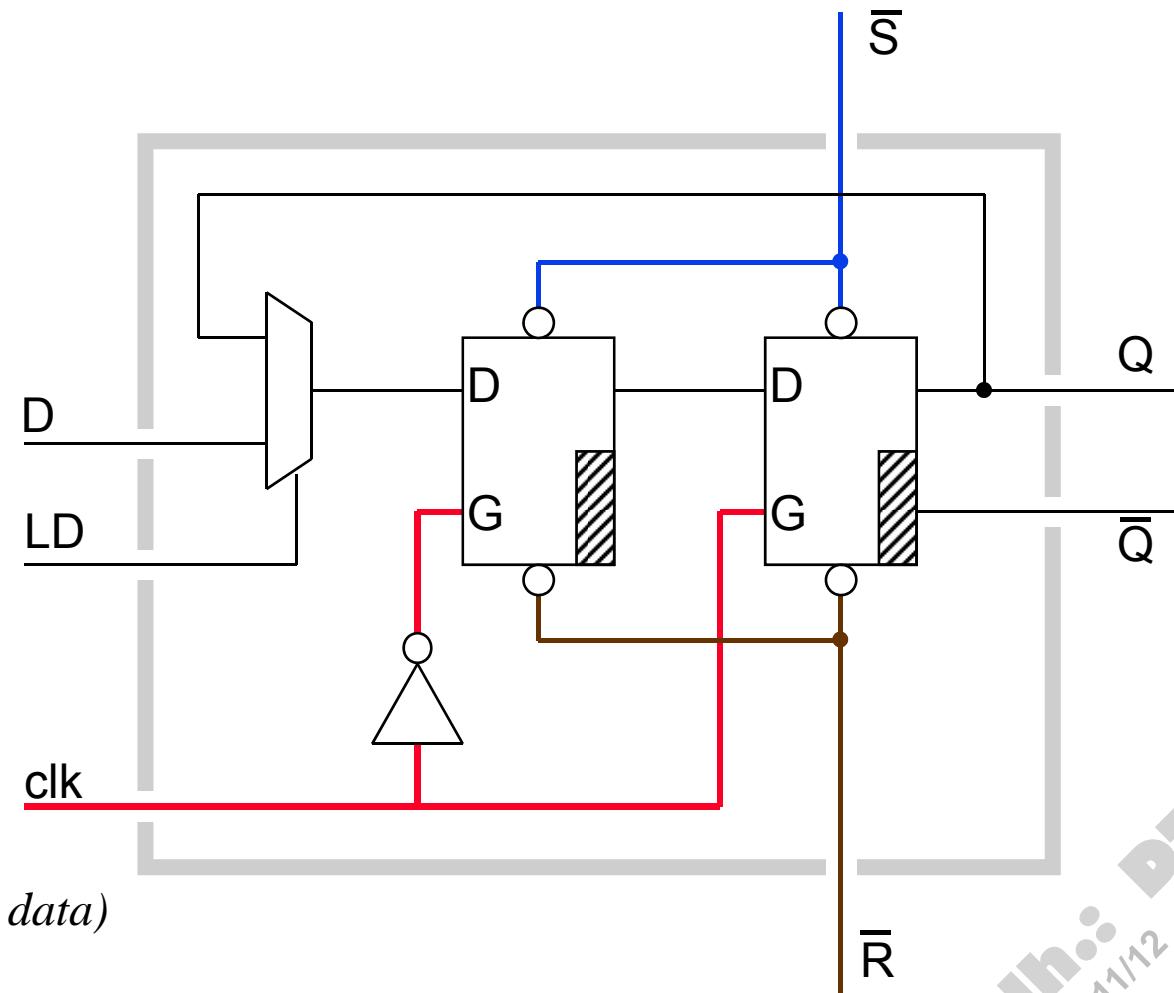
Laden



"Standard - ASIC"- FF

\bar{S}	\bar{R}	clk	LD	D	Q
1	1	1	1	x	D <i>load</i>
1	0	x	x	x	0
0	1	x	x	x	1
0	0	x	x	x	Pfui!!!
----- sonst -----					Q <i>store</i>

- D : data
- CE/LD : clk enable (*load FF with data*)
- R : reset
- S : (*pre-*)set
- clk : clock (*pulse*)



VHDL-Template

Register/FF mit asynchronem Reset

```
...
selbstErklärenderName:
process ( Takt & Reset ) is
begin
    if Reset then
        Initialisierung(en)
    elsif Taktflanke then
        Zuweisung(en)
    end if;
end process selbstErklärenderName;
...
```

Reset ist asynchron – der Reset wirkt sofort!

VHDL-Template

Register/FF mit synchronem Reset

...
(Signal-)Deklaration der Register-/FF-Signale **_cs** mit PowerUp-Initialisierung
...

...
selbstErklärenderName:
process (Takt) **is**
begin
 if Taktflanke **then**
 if Reset **then**
 Initialisierung(en)
 else
 Zuweisung(en)
 end if;
 end if;
end process selbstErklärenderName;
...

Reset ist synchron – wie alle anderen Wertänderungen erfolgt Reset synchron zu Taktflanke

Reset

- asynchroner Reset
 - (+) Power-Up
 - (-) synchrones Verlassen des Reset
- synchroner Reset
 - (-) Power-Up
 - (+) synchrones Verlassen des Reset
- Variante: Reset kommt asynchron und geht synchron
- Xilinx:
 - Kein Problem bei Power-Up
 - Bei Deklaration: `name_CS := initialValue;` initialisiert vor

Achtung! (1)

- Designs bestehen (bezüglich sequentieller Logik typischer Weise fast nur) aus FFs
- CMOS und nicht TTL ist die beherrschende Technik in ASICs
- Latches sind
 - eher eine „Ausnahme“
 - komplizierter (bezüglich einer fehlerfreien Ansteuerung in einem korrekten Design)
 - behindern Vereinfachungen (z.B. einfache Denkmodelle wie diskrete Zeit)
- (Nicht nur) Anfänger sollten **Latches** innerhalb eines Chips/FPGAs/CPLDs meiden

Wdh:: DT
WS11/12

Achtung! (2)

- **Keine sequentielle Logik (Zustände) selber bauen !**

Damit gleichbedeutend:

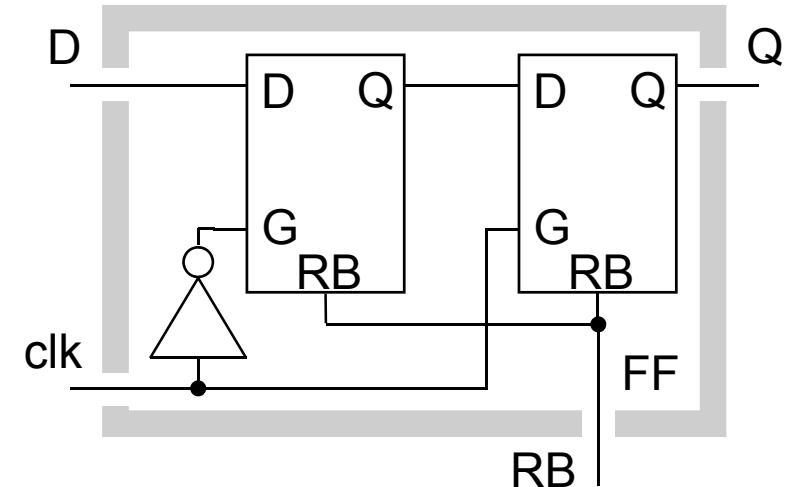
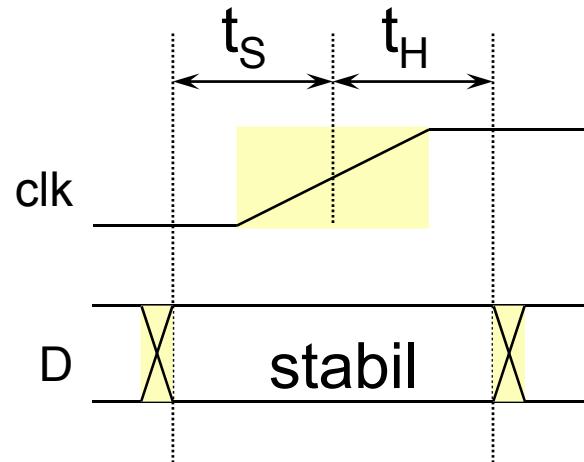
Keine Rückkopplungen in der kombinatorischen Logik.

Algorithmische Rückkopplungen laufen immer über Zustände/Sequentielle Logik.

- Immer auf existierende Zellen zurückgreifen
(bzw. der Synthese die Chance geben dies zu tun)

- Aber warum eigentlich?

Setup- und Hold-Time

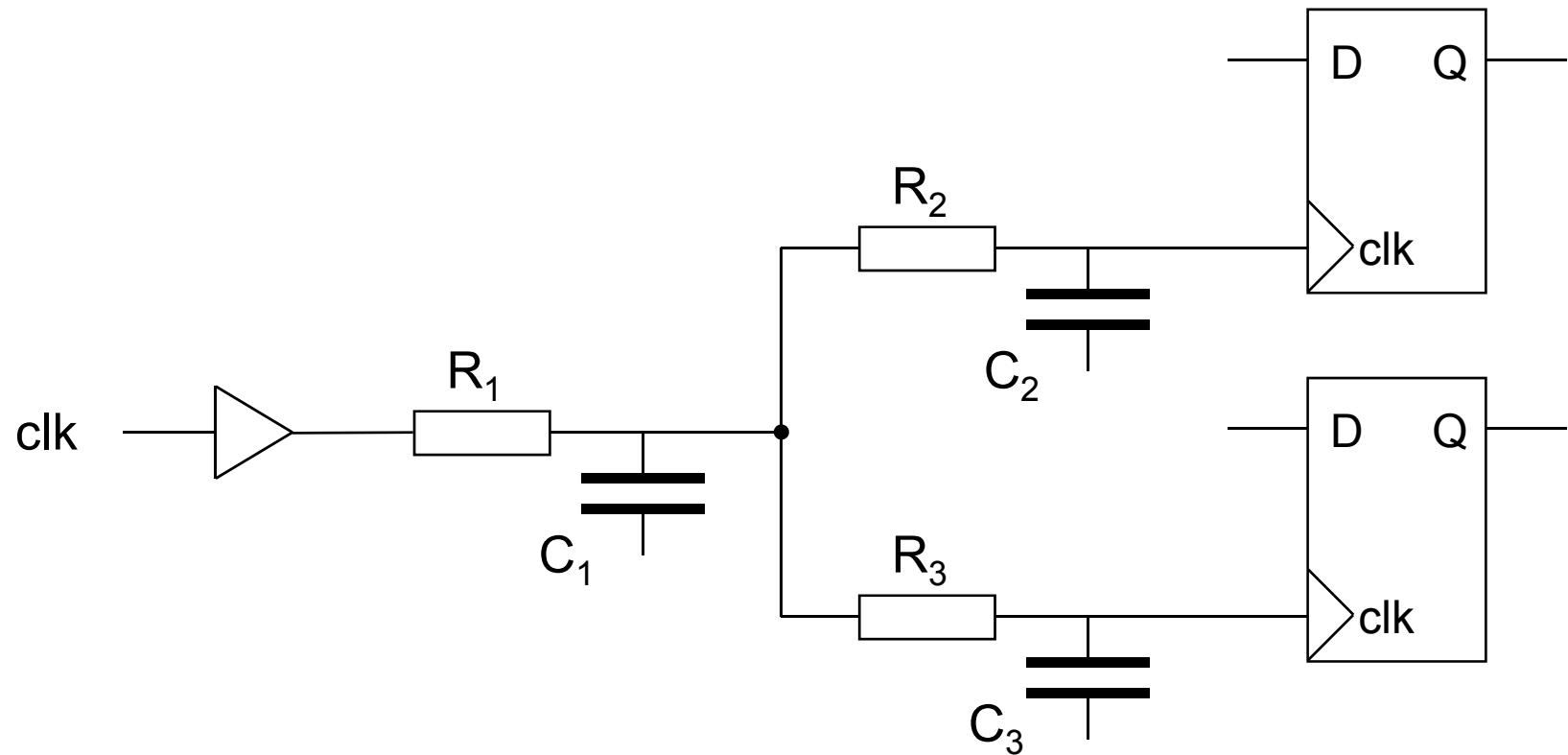


- Achtung! Bei der positiven Taktflanke (clk: 0->1) sind für einen kurzen Moment beide Latches offen ($G=1$). Änderungen am Dateneingang sind in diesem Moment kritisch. Flankensteilheit und Treiberstärke der Signale am Daten- und Takteingang beeinflussen das Ergebnis. Diese können von dynamischen Effekten abhängig sein
- Daten müssen eine gewisse Zeit vor der Taktflanke stabil sein (**setup time**) und dürfen sich auch nach der Taktflanke für eine gewisse Zeit nicht ändern (**hold time**)
- In einer Bibliothekszelle liegt die interne Logik extrem dicht beieinander. In einer selbstgebauten Zelle ist dies typischer Weise nicht der Fall.

"Vollsynchrones Design"

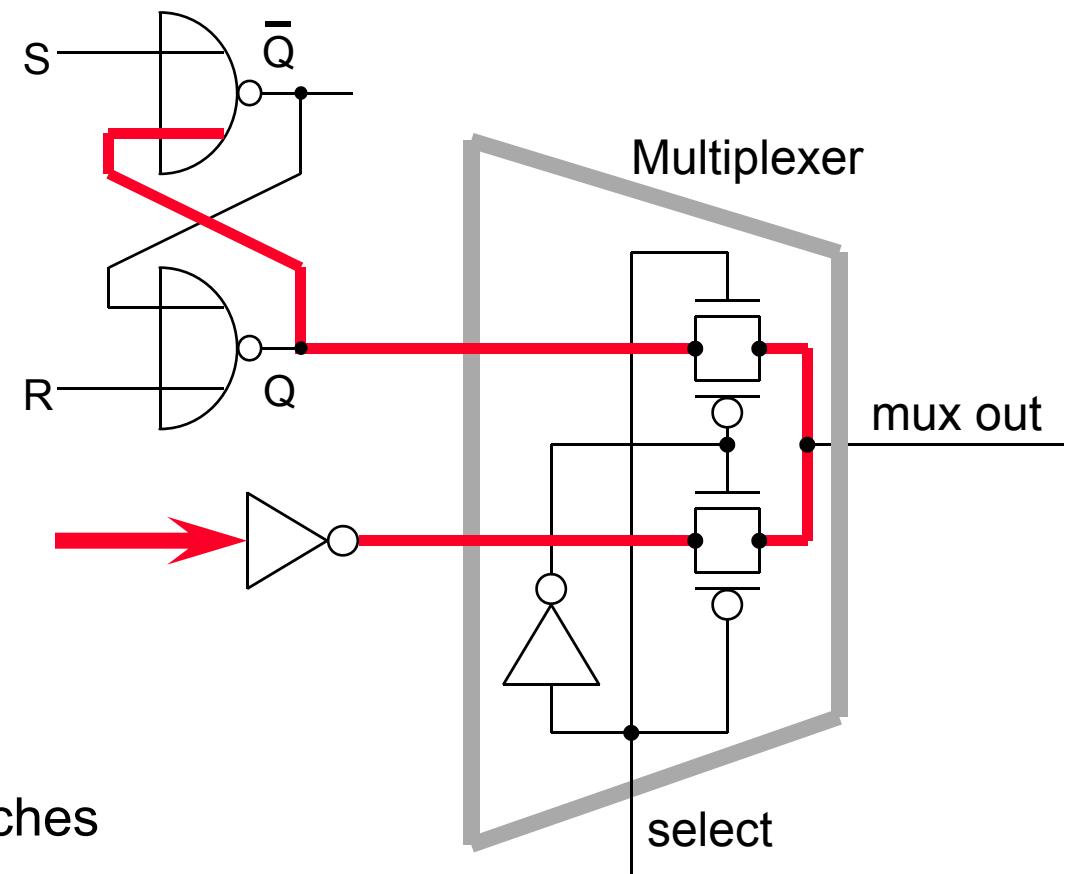
- Name bedeutet zunächst:
 - Es gibt FFs/Register und dieser "hängen" am Takt.
 - Alle Aktionen werden synchron zum Takt ausgeführt/angestartet.
- Im typischen Fall / Normalfall gilt:
 - Es gibt nur einen Takt
 - Alle FFs/Register hängen an diesem Takt
 - Alle FFs/Register arbeiten "positiv Flanken-gesteuert"
- Der entscheidende Punkt ist:
 - Aus einer "höheren" Abstraktionsschicht passiert die **Zustandsüberführung des "Systems"** zu einem Zeitpunkt.
- Aber wie schaut es in der Realität aus?

Clock-Skew



- **clk skew** liegt vor, wenn die Taktflanken zu **unterschiedlichen Zeitpunkten** an den Flipflops eintreffen
- damit Tools den clk skew minimieren können müssen sie die FFs (er)kennen.
Bibliothekszellen sind leicht zu erkennen

Vermeide selbstgebaute Zustände



selbstgebaute Flipflops oder Latches

- können unerwartetes Verhalten zeigen
- werden nicht von allen Tools beherrscht
- müssen unbedingt vermieden werden;
unbedingt **fertige** Bibliotheks-Zellen verwenden