HOCHSCHULE KONSTANZ
FAKULTÄT INFORMATIK

Hardware/Software-Codesign

Vorlesung
Prof. Dr.-Ing. I. Schoppa

Der vorliegende Text ist ein unvollständiges Begleitmaterial,
das zum Gebrauch in der Vorlesung und in der Übung bestimmt ist.

- Prof. Dr. Irenäus Schoppa
- Fakultät Informatik
- Gebäude F, Zimmer F124
- Telefon: 07531 / 206 – 644
- E-Mail: irenaeus.schoppa@htwg-konstanz.de
- Sprechstunden nach Vereinbarung

- Unterlagen:
  \\merkur\Lehre\Schoppa\HSCD\Vorlesung
  \\merkur\Lehre\Schoppa\HSCD\Uebung\Aufgabe

Organisation

Zeit	Mo	Di	Mi	Do	Fr
08:00 09:30				VL HSCD F-130	
09:45 11:15		UE HSCD F-130			
11:30 13:00					
14:00 15:30					
15:45 17:15					

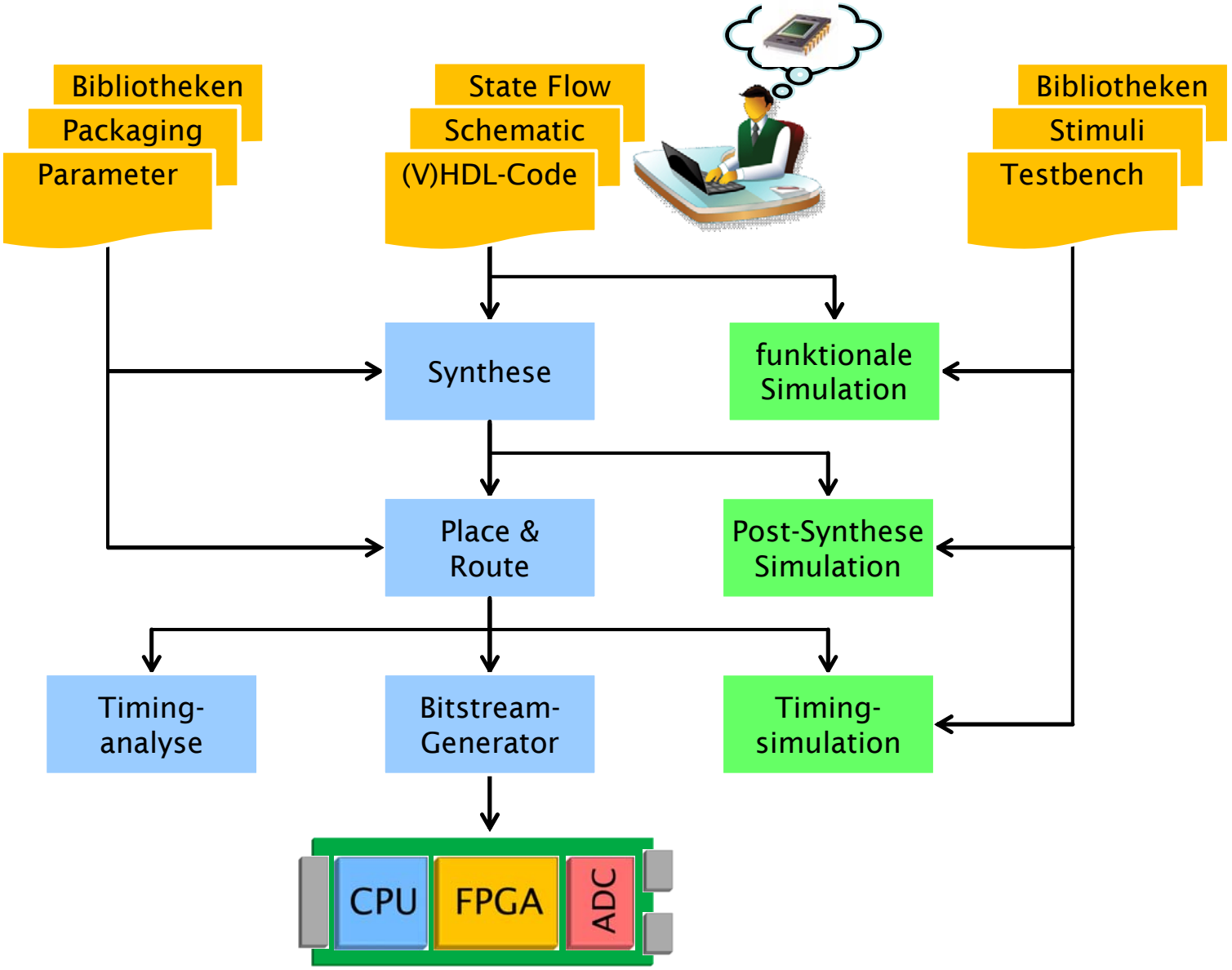
Literatur

1. Xilinx: *PicoBlaze 8-bit Embedded Microcontroller User Guide for Spartan-3, Spartan-6, Virtex-5 and Virtex-6 FPGAs*, Xilinx Corp. 2010. (ug129.pdf)
2. K. Chapman: *KCPSM3 8-Bit Micro Controller for Spartan-3, Virtex-II and Virtex-IIPro*, Xilinx Corp. 2003. (kcpsm3_manual.pdf)
3. Xilinx: *Library Guide*, Xilinx Corp. 2005. (libg.pdf)
4. K. Skahill: *VHDL for Programmable Logic*, Addison-Wesley, 1994.
5. Actel: *Actel HDL Coding - Style Guide*, Actel Corporation, 2005. (hdlcode.pdf)
6. G. Lehmann, B. Wunder und M. Selz: *Schaltungsdesign mit VHDL*, Franzis Verlag, 1994. (Schaltungsdesign.pdf)
7. J. Reichardt und B. Schwarz: *VHDL-Synthese: Entwurf digitaler Schaltungen und Systeme*, Oldenbourg, 4. Auflage, 2007.
8. Xilinx: *Benutzerhandbücher*, *Datenblätter*, *Applikationsberichte*, Xilinx Corporation, 2007..2010.

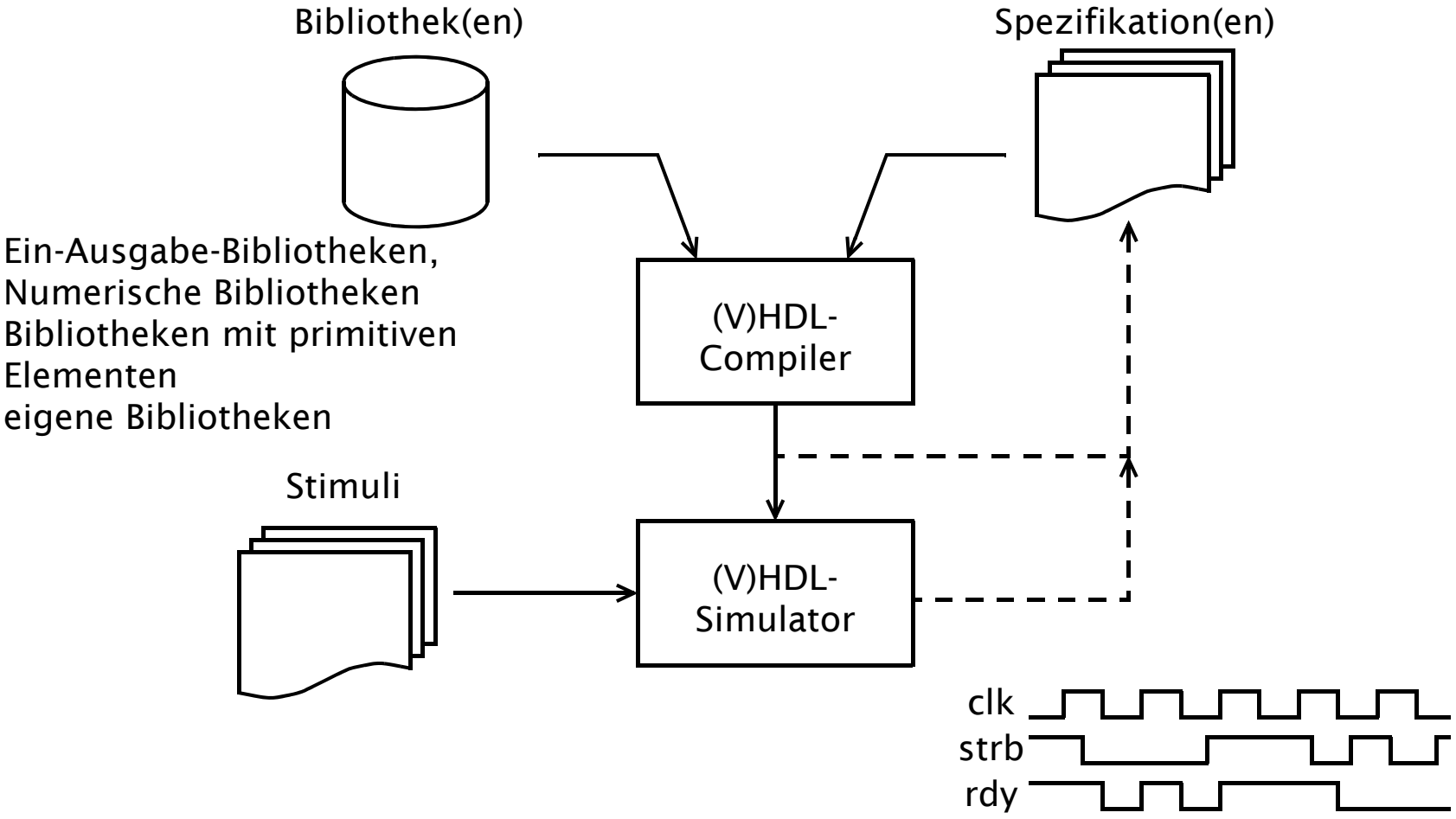
Themenübersicht

- Soft-Core-Prozessor *PicoBlaze*
- Hardware-/Software Codesign
- Einführung in VHDL
- Synthese applikationsspezifischer (Co-)Prozessoren
  - Interface- und Kommunikationssynthese
  - Datenpfad- und Steuerpfadsynthese
  - Datenabhängigkeiten
  - Ablaufplanungsalgorithmen
- Direkte Hardware-Compilation

Design Flow

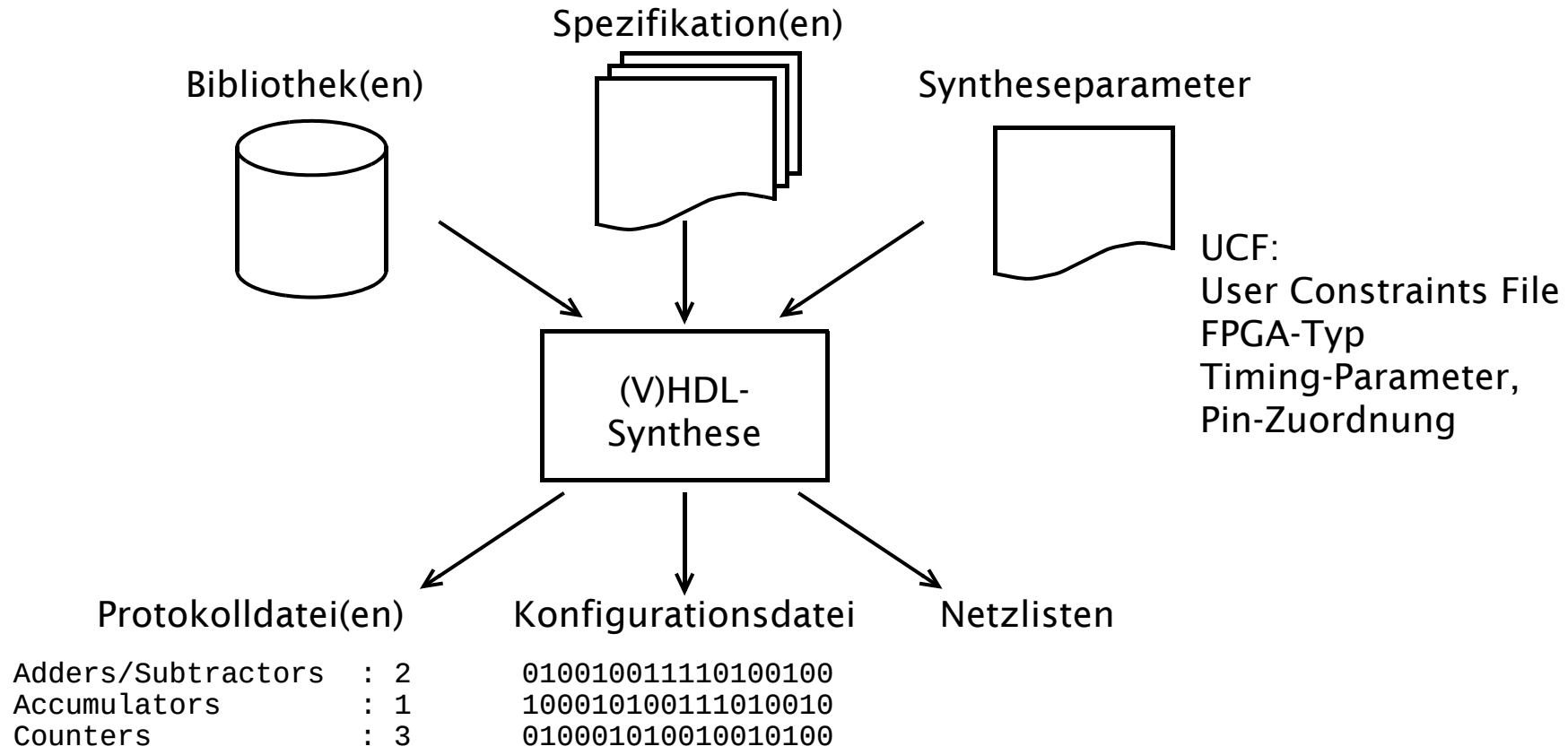

Bibliotheken
Packaging
Parameter
State Flow
Schematic
(V)HDL-Code
Bibliotheken
Stimuli
Testbench
Synthese
funktionale Simulation
Place & Route
Post-Synthese Simulation
Timing-analyse
Bitstream-Generator
Timing-simulation
CPU
FPGA
ADC

- Rechnergestützter Entwurf: (V)HDL-Simulation



Einführung

- Rechnergestützter Entwurf: (V)HDL-Synthese

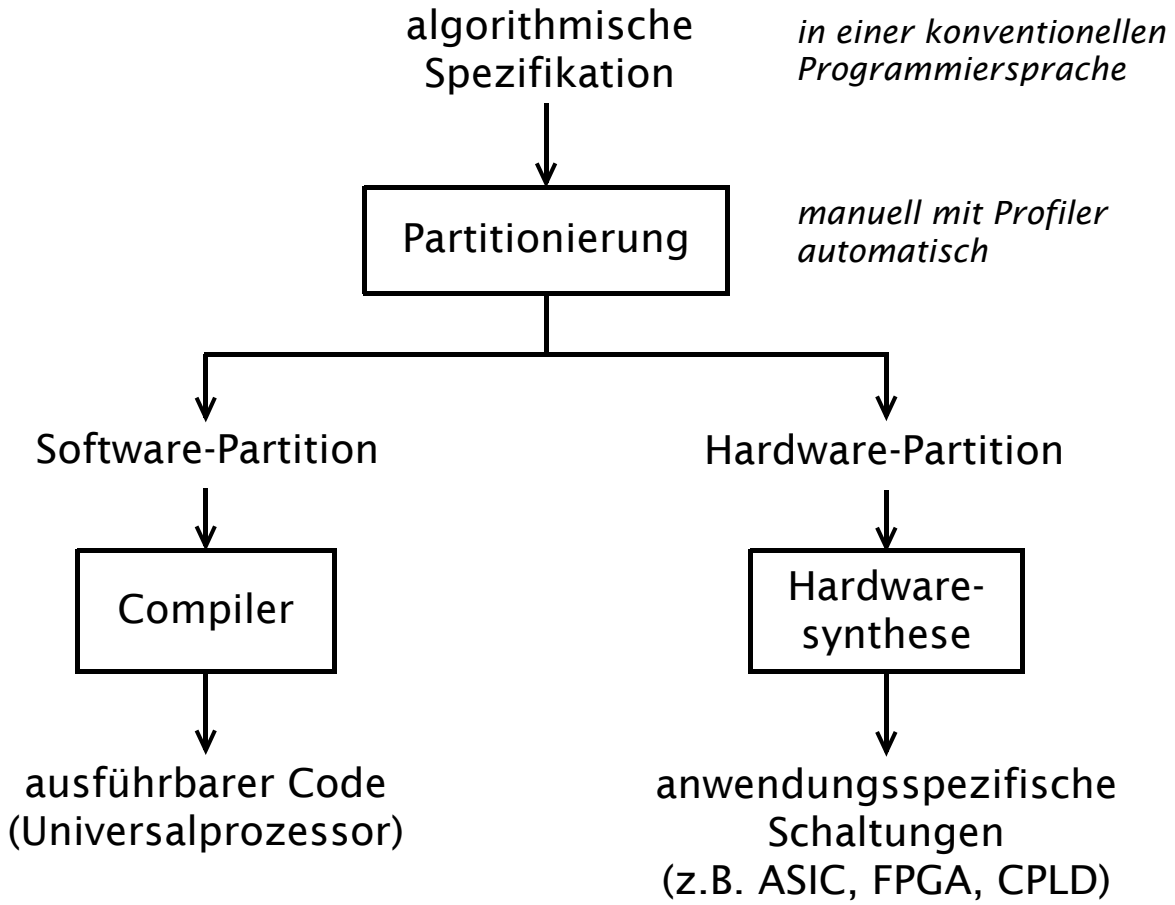


- 90-10-Regel
  90% der Ausführungszeit in 10% des Programmcodes
  - die meiste Laufzeit wird in einem oder in einigen relativ kleinen Abschnitten eines Programms verbracht.
  - Werden diese 10% durch spezielle Hardware – z.B. als applikations-/kundenspezifische Schaltungen – realisiert, und nimmt man an, daß die Ausführung der in Hardware realisierten Abschnitte nur halb so lange dauert wie auf einem Universalprozessor, so läßt sich die Gesamtausführungszeit des Programms theoretisch auf 10% + 90%/2 = 55% der ursprünglichen Zeit reduzieren.
  - Die effektive Gesamtausführungszeit verlängert sich in der Regel um zusätzlichen Kommunikationsaufwand zwischen den in Hard-ware und den in Software realisierten Programmabschnitten.

Hardware/Software-Codesign

- Der Hardware- und Softwareentwurf werden als ein Gesamtproblem betrachtet, bei dem eine abstrakte algorithmische Problemspezifikation in eine Implementierung transformiert wird, die aus Hardware- und Softwarekomponenten besteht.
- Zielsystem: eine Systemarchitektur, die auf einem Universalprozessor mit Speicher und Peripheriebausteinen basiert und um zusätzliche applikationsspezifische Hardwarekomponenten erweitert ist.
- Berechnungsintensive und zeitkritische Teile einer algorithmischen Spezifikation werden auf speziellen (applikationsspezifischen) Hardwarekomponenten ausgeführt.
- Nichtkritische Anwendungsteile sowie Betriebssystemfunktionen werden weiterhin auf einem Universalprozessor ausgeführt.

Hardware/Software-Codesign



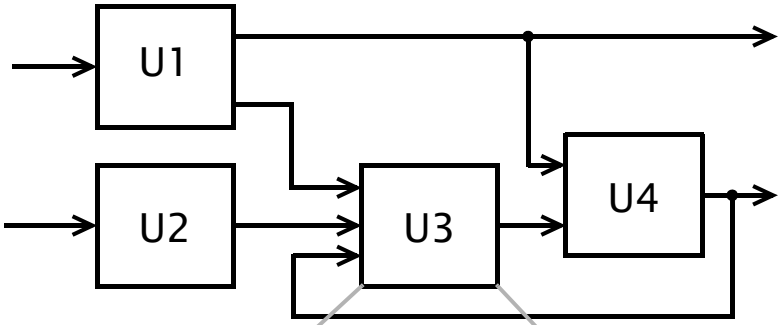
Hardware/Software-Codesign

- Eine homogene Gesamtbeschreibung eines Systems auf einer hohen Abstraktionsebene.
- Der Anwender braucht weder eine neue HDL lernen noch über umfassende Hardwarekenntnisse verfügen.
- Während der Implementierung eines Algorithmus braucht der Anwender nicht auf die ihm vertraute Entwicklungs-umgebung und bewährte Testmethoden (z.B. Debugger) verzichten.
- Für den funktionalen Test der Spezifikation ist keine besondere (laufzeitintensive) Simulation erforderlich. Statt-dessen wird diese Spezifikation nach einer Übersetzung mit einem Compiler auf einem Allzweckrechner ausgeführt.

Hardware/Software-Codesign

- Hauptaufgabe im Hardware/Software-Codesign:
  - Codesegmente in Programmen unter Berücksichtigung von Randbedingungen, z.B. Echtzeit-/Leistungsanforderungen, zu identifizieren, die für eine Hardwarerealisierung geeignet sind.
  - Software- und hardwareorientierte Ansätze:
    - softwareorientierter Ansatz geht von einer Implementierung eines Systems als reine Softwarelösung aus. Sukzessiv werden Codesegmente in Hardware verlagert, bis vorgegebene Randbedingungen erfüllt sind.
    - hardwareorientierter Ansatz geht davon aus, daß das gesamte System anfangs in Hardware realisiert wird. Solange vorgegebene Randbedingungen erfüllt sind, werden Codesegmente in Software transferiert.
- Analyse eines Programms:
  - manuelle Laufzeitanalyse mit einem Profiler
  - automatische Untersuchung statischer und dynamischer Eigenschaften einzelner Softwaremodule

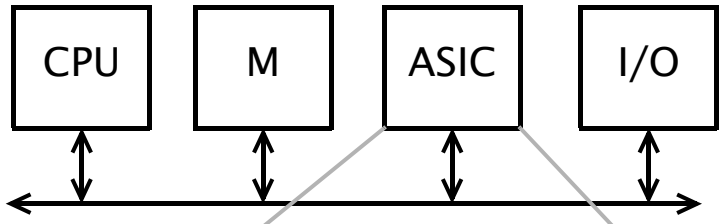
- Architekturen mit appl.-spezifischen Netzwerktopologien:
  - Hardwarekomponenten sind so miteinander verdrahtet, wie dies zur effizienten Lösung einer bestimmten Aufgabe notwendig ist.
    - speziellen Anwendungen: z.B. Algorithmen der digitalen Bild-/Signalverarbeitung
    - Erweiterung umständlich, oft ein Redesign des Systems notwendig.
  - Schnittstelle einer Komponente:
    - die Anzahl der Parameter,
    - deren Datentyp (z.B. Gleitkommazahlen, ASCII-Zeichen) und
    - die Richtung des Informationsflusses (Ein-/Ausgabe)



Beispiel einer fiktiven Schaltung

```
entity asic is
  port(
    reset: in  bit;
    clk:   in  bit;
    start: in  bit;
    ready: out bit;
    in1:   in  bit_vector(0 to 15);
    in2:   in  bit_vector(0 to 15);
    in3:   in  bit_vector(0 to 15);
    res:   out bit_vector(0 to 15)
  );
end asic;
```

- Systemarchitekturen mit einem zentralen Systembus
  - basiert auf einem Universalprozessor mit Speicher-, Peripheriebausteinen und speziellen Systemkomponenten:
    - gemeisamer, zentraler Systembus (nachteilig: zeitsequentielle Datentransporte, Informationsaustausch zu einem Zeitpunkt nur zwischen zwei Systemkomponenten)
    - Flexibilität und Universalität: Erweiterung um zusätzliche Systemkomponenten relativ einfach
  - Schnittstelle einer Komponente:
    - vorgegeben durch die Eigenschaften des Systembusses
    - Daten- und Adreßbusbreite
    - Handshake-Signale fürs Busprotokoll



Beispiel einer fiktiven Schaltung

```
entity asic is
  port(
    reset: in    bit;
    clk:   in    bit;
    ce:    in    bit;
    rw:    in    bit;
    int:   out   bit;
    dbus:  inout bit_vector(0 to 7);
    abus:  in    bit_vector(0 to 3)
  );
end asic;
```

Hardware/Software-Codesign

```
#define STRT 0x01
#define EOP  0x04

#define ADR 0xFF00

typedef struct {
  int x, y;
  char scr;
} ass_reg;
```

```
int res;
...
volatile ass_reg *ass = (ass_reg *)ADR;

ass->x = ...;
ass->y = ...;                  Parameterübergabe

ass->scr = STRT;               Ausführung starten

while (!(ass->scr & EOP)) ;    Synchronisation

res = ass->x;                  Ergebnis übernehmen
```

```
int x, y;

void gcd (void) {
  while (x != y)
    if (x < y)
      y = y - x;
    else
      x = x - y;
}
```