TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Technische Informatik, Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur

Vortrag zum Hauptseminar „Hardware/Software Co-Design“

Robert Mißbach

Dresden, 02.07.2008

Gliederung

1 Einleitung

Was ist HW/SW Co-Design?

- stark verkoppelte, zeitgleiche Entwicklung von Hardware- und Softwarekomponenten eines digitalen Systems
  - Analyse von Schnittstellen zwischen Hardware und Software
  - Bewertung von Entwurfsalternativen

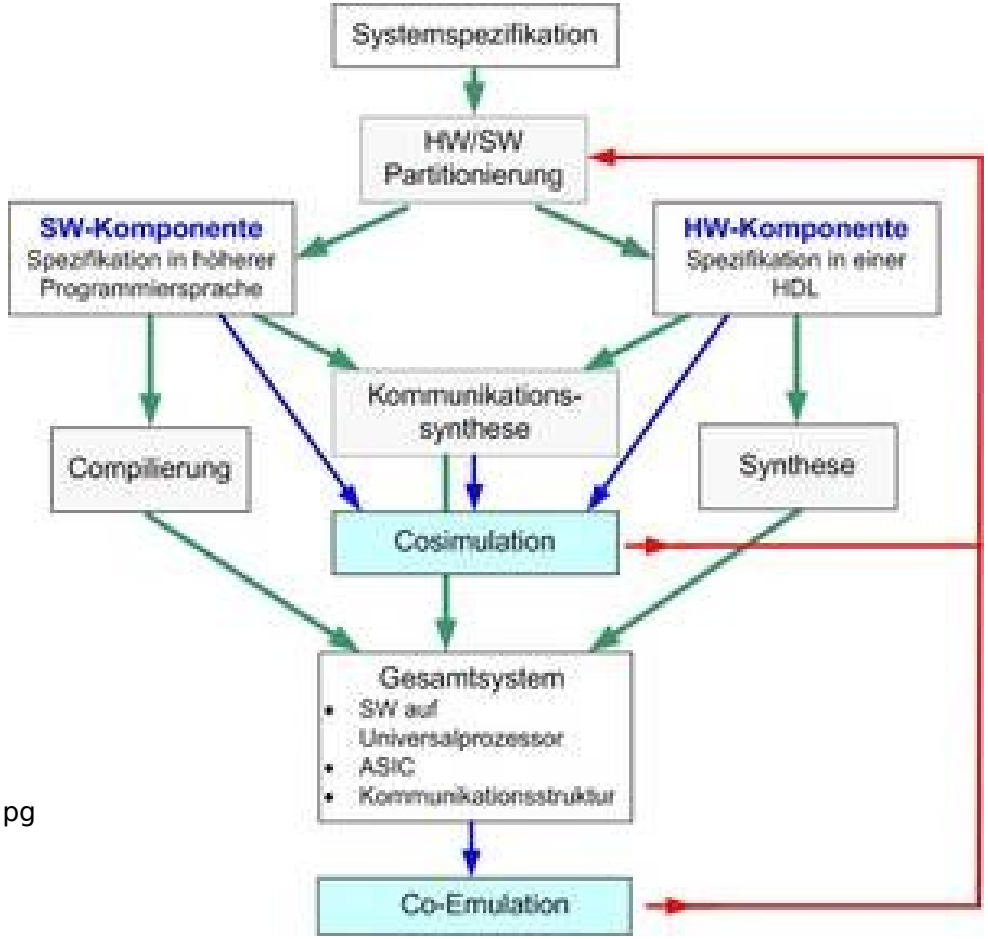
1 Einleitung

Motivation für Co-Design

- Design moderner eingebetteter Systeme muss optimiert werden
  - komplexe Systeme, hohe Flexibilität
  - heterogene Zielsysteme
    - Prozessoren, ASICS, FPGAs, DSP, ...
  - meist Einhaltung bestimmter Randbedingungen nötig
    - Performance, Kosten, Stromverbrauch, Zuverlässigkeit, ...
- Reduktion von Kosten und time-to-market

1 Einleitung

Designflow



Quelle: http://www.iti.uni-luebeck.de/typo3temp/pics/ce4e332b4f.jpg

2 Systemspezifikation

Ziele

- Vollständige formale Beschreibung des kompletten Systems
- Verwendung des Modells für Verifikation oder Synthese
- Noch keine Aussage über Aufteilung in Hard- und Softwarekomponenten

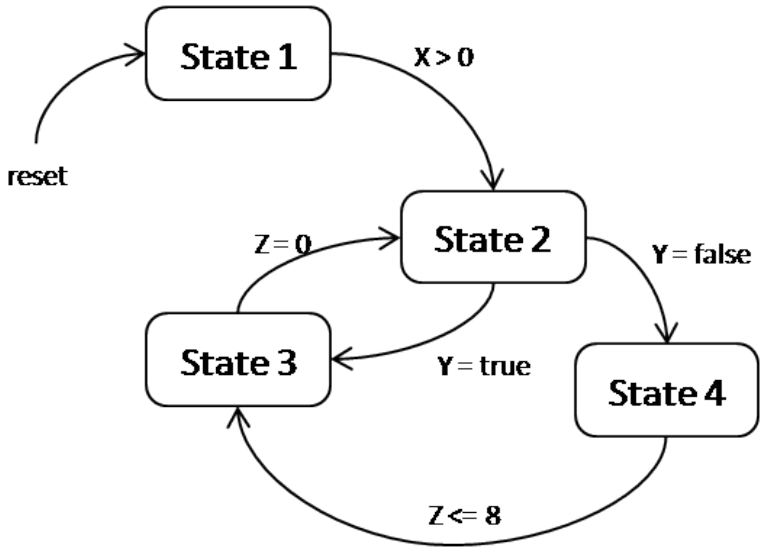
2 Systemspezifikation

Anforderungen

- Modellierung von Hierarchien
- Modellierung von Zeitverhalten
- Modellierung von Zuständen und Zustandsübergängen
- Modellierung von Datenflusskonzepten

2 Systemspezifikation

Finite State Machines

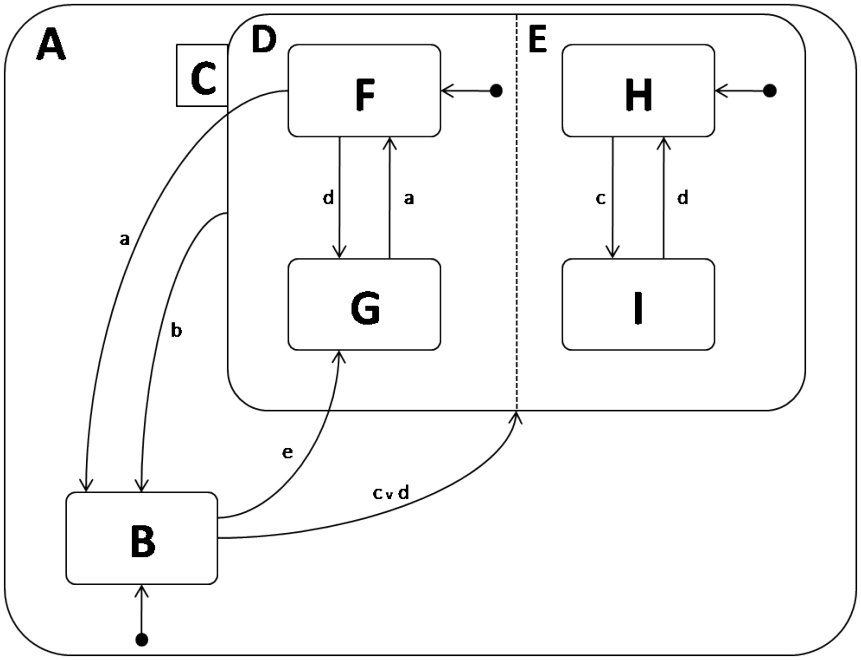


- Extrem unübersichtlich bei komplexeren Systemen

→ Weiterentwicklung zu StateCharts

2 Systemspezifikation

StateCharts



- Vorteile:
  - Hierarchie
  - zahlreiche Simulationstools erhältlich
  - backends zur Übersetzung in C- oder VHDL-Code
- Nachteile:
  - generierte C-Programme meist nicht besonders effizient
  - keine Beschreibung von struktureller Hierarchie
  - nicht geeignet zur Modellierung verteilter Systeme

2 Systemspezifikation

weitere Möglichkeiten der Modellierung

- Esterel
- SDL
- Petri-Netze
- ...

3 Partitionierung

Trade-offs

- Verschiebung von Funktionalität zur Software
  - geringere Stückkosten
  - bessere Wartbarkeit/Änderbarkeit
  - Performance-Einbußen
- Verschiebung von Funktionalität zur Hardware
  - höhere Stückkosten
  - meist deutlicher Performance-Gewinn
  - Leistungsaufnahme kann gezielt reduziert werden

3 Partitionierung

Partitionierungsproblem

- Zuordnung von n Objekten der Objektmenge O = {o_1, o_2, ..., o_n} zu einer von m Partitionen aus der Menge P = {p_1, p_2, ..., p_m}, so dass gilt:
  - $p_1 \cup p_2 \cup ... \cup p_m = O$
  - $p_i \cap p_j = \emptyset \quad \forall i, j : i \neq j$
  - $Kosten \quad c(P) \quad minimal$

3 Partitionierung

Kostenfunktion

- Definition einer Kostenfunktion, um Qualität eines Systems zu bestimmen
- Beispiel:

$$f(C,L,P) = k_1 \cdot h_c(C, C_{max}) + k_2 \cdot h_L(L, L_{max}) + k_3 \cdot h_P(P, P_{max})$$

C	Systemkosten
L	Latenz
P	Leistungsaufnahme
h_C, h_L, h_P	gibt an, wie stark C, L und P von den Design Constraints C_{max}, L_{max} und P_{max} abweichen
k_1, k_2, k_3	Gewichtung

3 Partitionierung

Greedy-Algorithmus

- Bipartitionierungsproblem: $P = \{p_{SW}, p_{HW}\}$
- Initialisierung z.B. mit $P = \{O, \{\}\}$
  - Alle Funktionen in Software realisiert
  - Migration von bestimmten Funktion in die Hardware

3 Partitionierung

Greedy-Algorithmus

- Migrieren von Objekten in die jeweils andere Partition, bis keine Verbesserungen mehr eintreten

```
REPEAT{
    P_old = P;
    FOR i=1 TO n {
        IF (f(Move(P, o_i)) < f(P)) {
            P = Move(P, o_i);
        }
    }
UNTIL (P == P_old)
```

3 Partitionierung

weitere Partitionierungsalgorithmen

- Kernighan – Lin Algorithmus
- Simulated Annealing
- Integer Linear Programming (ILP)
- evolutionäre Algorithmen
- ...

4 Systemverifikation

Cosimulation

- Untersuchung des spezifizierten Systems
  - Beobachtung des Systemverhaltens über einen bestimmten Zeitraum
  - Änderung von bestimmten Design-Parameter (Design Space Exploration)
- System Debugging

4 Systemverifikation

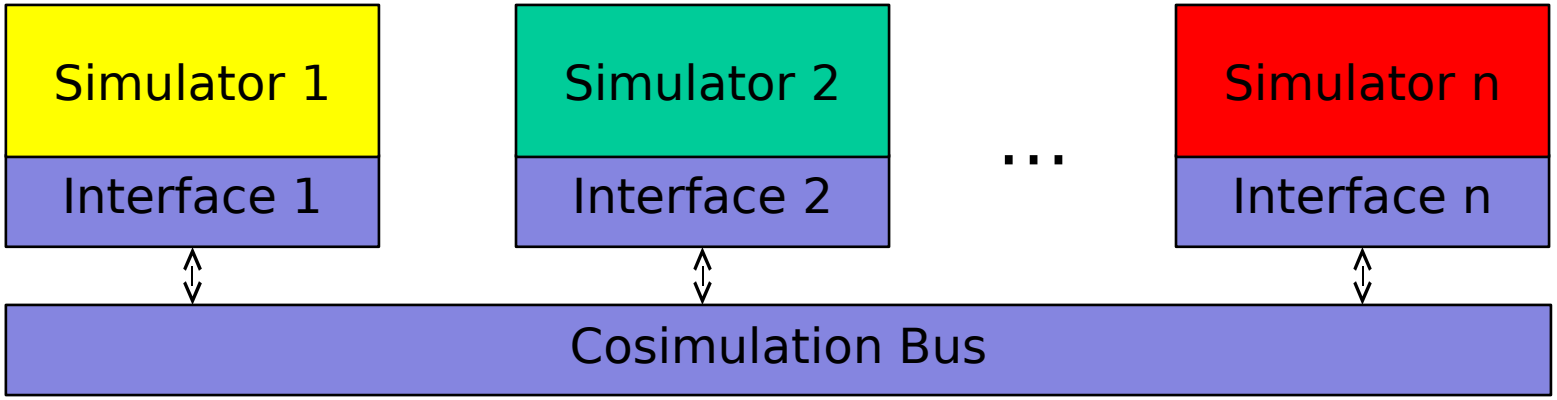
Cosimulation

- Ziel: Simulation des ganzen Systems, bestehend aus unterschiedlichen Modulen

- Probleme:
  - unterschiedliche Grade der Abstraktion
  - verschiedene Simulatoren nötig
  - unterschiedliche Kommunikations-Protokolle
  - Synchronisation muss gewährleistet werden

4 Systemverifikation

Cosimulation



- Simulationsumgebung wird durch CAD-Tools bereitgestellt

4 Systemverifikation

Rapid Prototyping

- Schnelle Erstellung eines lauffähigen Prototypen

- Ziel:
  - Überblick über Einhaltung von Randbedingungen
  - Schnelle Erkennung von Problemen beim Einsatz des Systems
  - Erkennung von Problemen im Konzept

Quellen

- http://www.tik.ee.ethz.ch/tik/education/lectures/hswcd/
- http://www12.informatik.uni-erlangen.de/edu/hscd/script/
- http://www.mr.inf.tu-dresden.de/?ln=de&tl=lehre_ws&sl=es&bl=ws0708&site=ws0708
- http://www-ihs.theoinf.tu-ilmenau.de/lehre/hs/HS%20Rossa%20und%20Dingler/Vortrag_HS_IHS.pdf
- http://mint.cs.man.ac.uk/Projects/UPC/Reviews/Cosimulation.html
- http://www.iti.uni-luebeck.de/index.php?id=hd_co-design&L=1

- Giovanni De Micheli, Mariagiovanna Sami: Hardware/Software Co-Design, Kluwer Academic Publishers, 1996