Hardware Operating Systems

Ulrich Langenbach

Institut für Robotik und Prozessdatenverarbeitung
Technische Universität Berlin

Dynamische Rekonfiguration Eingebetteter Systeme
PDV Seminar SS06

Gliederung

Motivation

Hardware Operating Systems (HOS)

- Produktivitätssteigerung
- erhöhte Portabilität
- vereinfachte Systemrepartitionierung
- vereinfachtes Debugging

Definition Betriebssystem

Betriebssystem (Definition nach DIN 44300)

Die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften der Rechenanlage die Grundlage der möglichen Betriebsarten des digitalen Rechensystems bilden und insbesondere die Ausführung von Programmen steuern und überwachen.

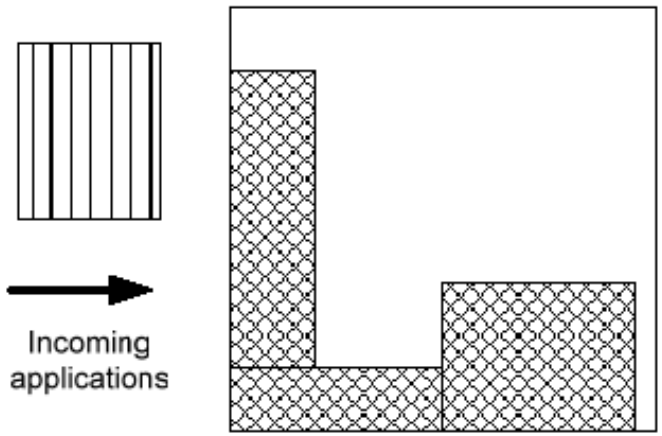
Aufgaben traditioneller Betriebssysteme

- Speicherverwaltung
- Prozessverwaltung
- Scheduling
- Hardwareverwaltung
- Zeitdienste
- Synchronisationsdienste
- Kommunikation
- Abstraktion

Aufgaben Hardwarebetriebssystem

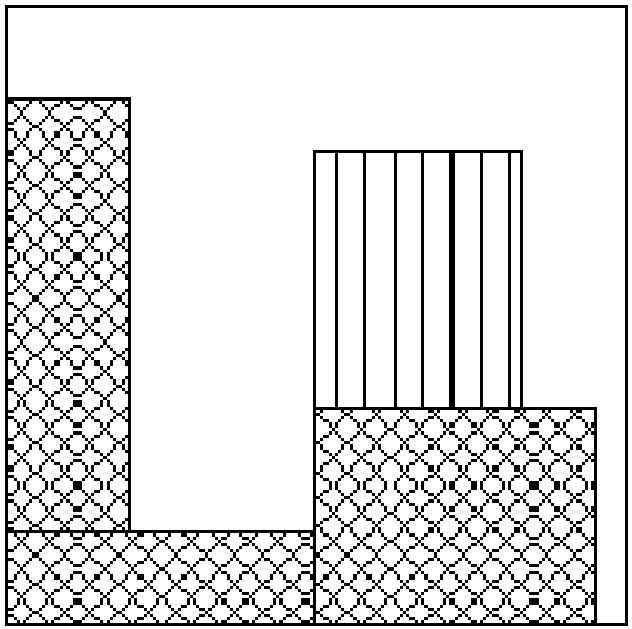
- → Platzierung
- → Flächenverwaltung im FPGA
- → (Partitionierung der rekonfigurierbaren Hardware)
- → (Routing)
- → Kommunikation
- → Speicherverwaltung (auch Spezialblöcke)
- Scheduling
- Synchronisation
- Rekonfiguration

Platzierung der Hardware im FPGA (1/2)



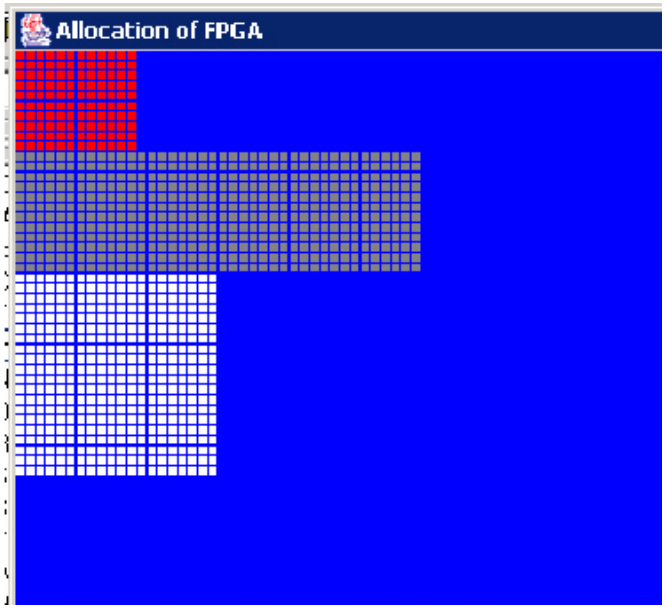
- NP hartes Problem
  $\rightarrow$ Heuristiken
- Ziel, Minimierung externer Fragmentierung
- Algorithmus muss schnell sein

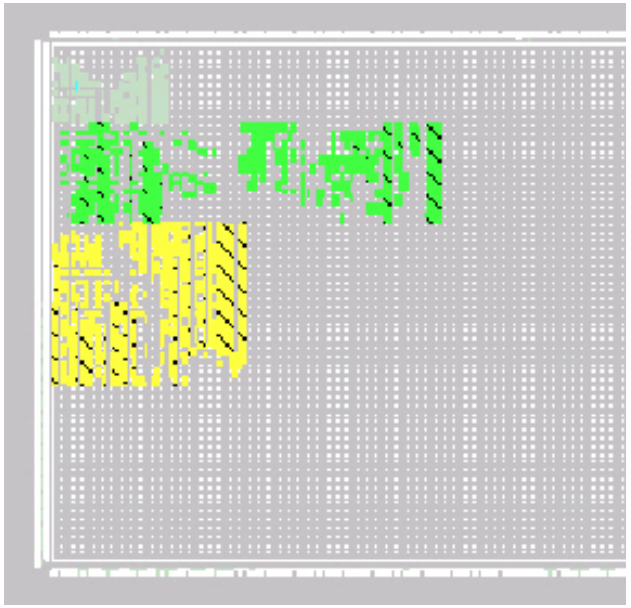
Platzierung der Hardware im FPGA (2/2)



- bin packing Problem
  - genetische Algorithmen
  - Simulated Annealing
- dynamisches und statisches Scheduling nötig
- $\rightarrow$ partitioning in cluster computing

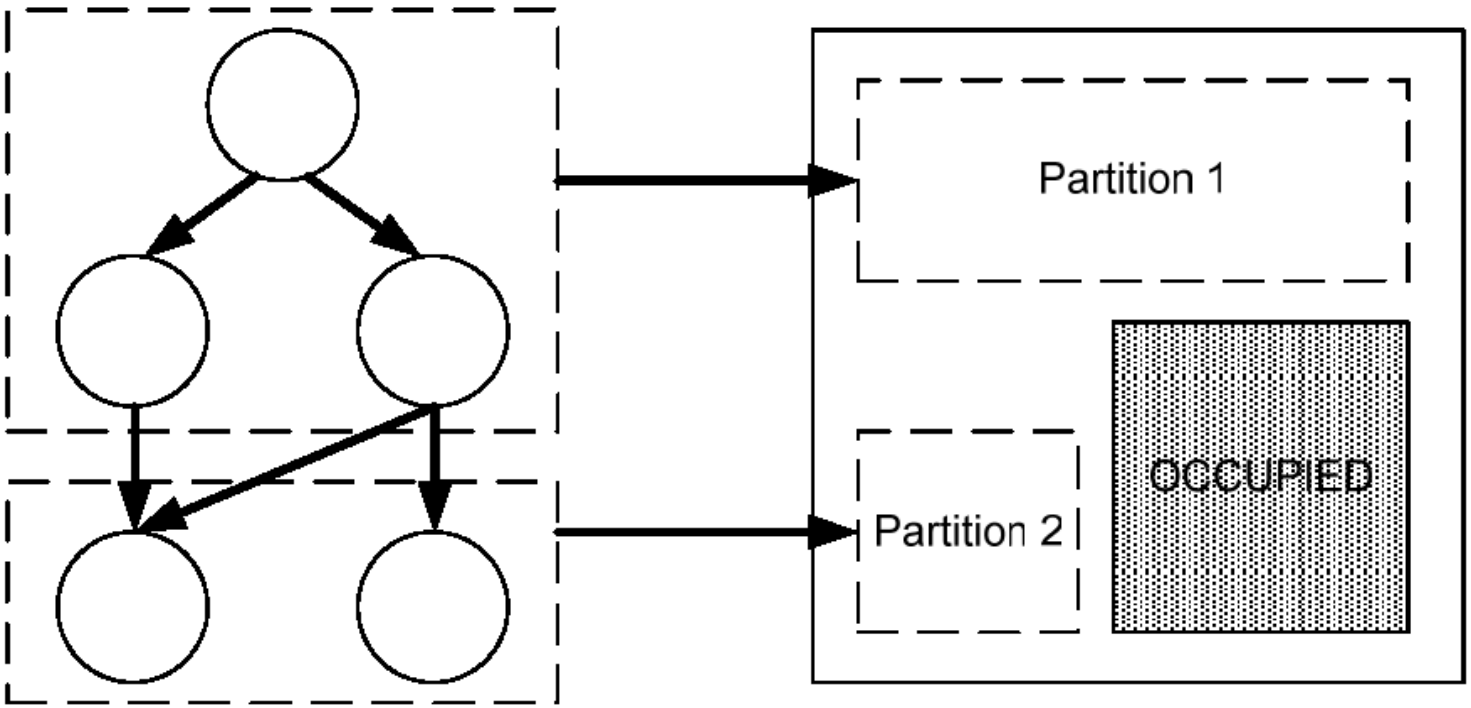
Fragmentierung





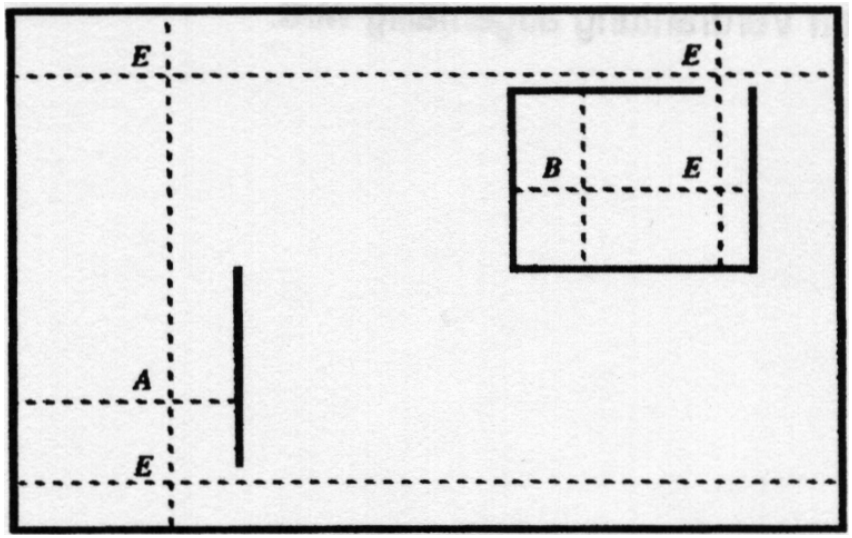
- interne Fragmentierung
- externe Fragmentierung

Taskpartitionierung



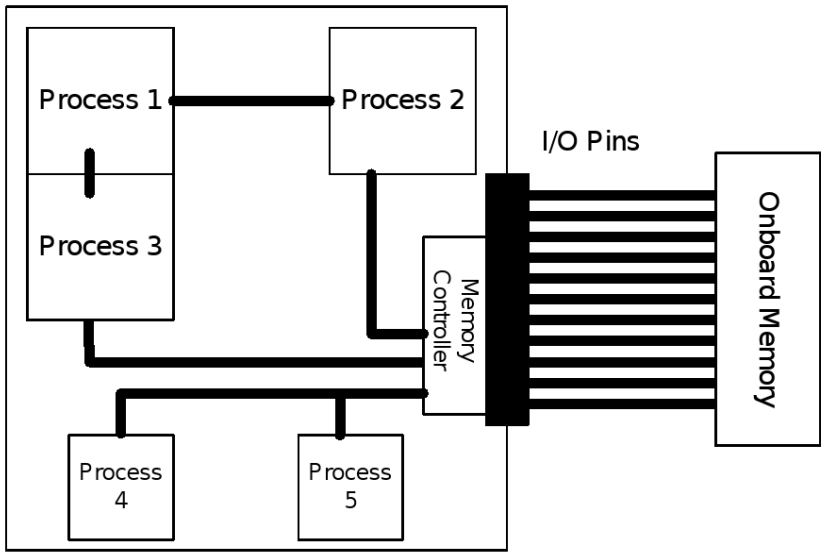
- erfordert Taskbeschreibung als SDF
- erfordert Onlinesynthese

Routing im FPGA



- NP hartes Problem $\rightarrow$ Heuristiken
- Algorithmus muss schnell sein
- rel. einfach, da lokale Beziehungen vorherrschen

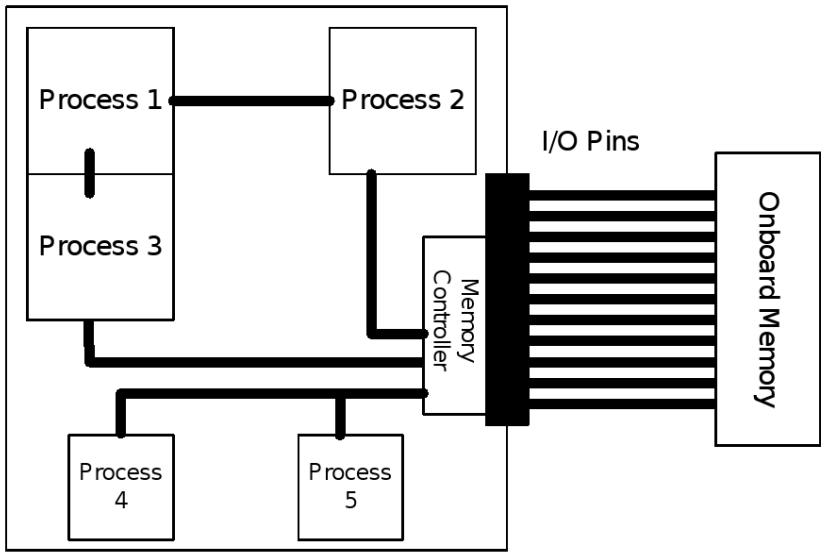
Kommunikationsinfrastruktur



Busse

- relativ einfach und bekannt
- es können Engpässe auf dem Bus entstehen
- relativ flexibel

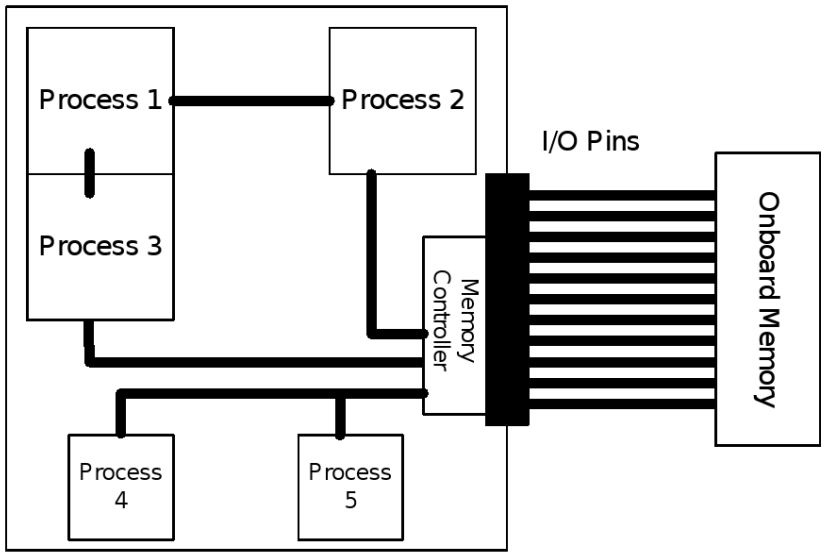
Kommunikationsinfrastruktur



NoCs

- größerer Overhead durch Switchmodule
- flexibel
- Prinzipien verstanden, Techniken neu

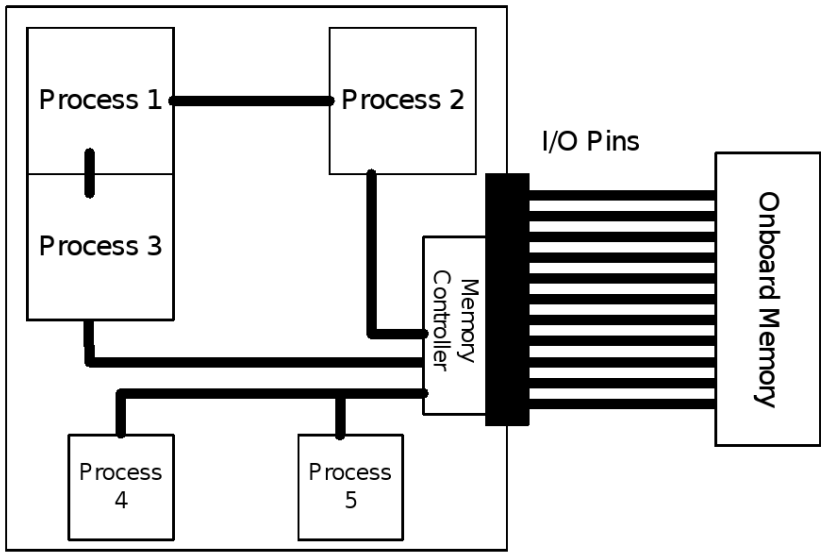
Kommunikationsinfrastruktur



direkte Verbindung

- höchste Parallelität
- unflexibel
- impliziert dynamisches Routing

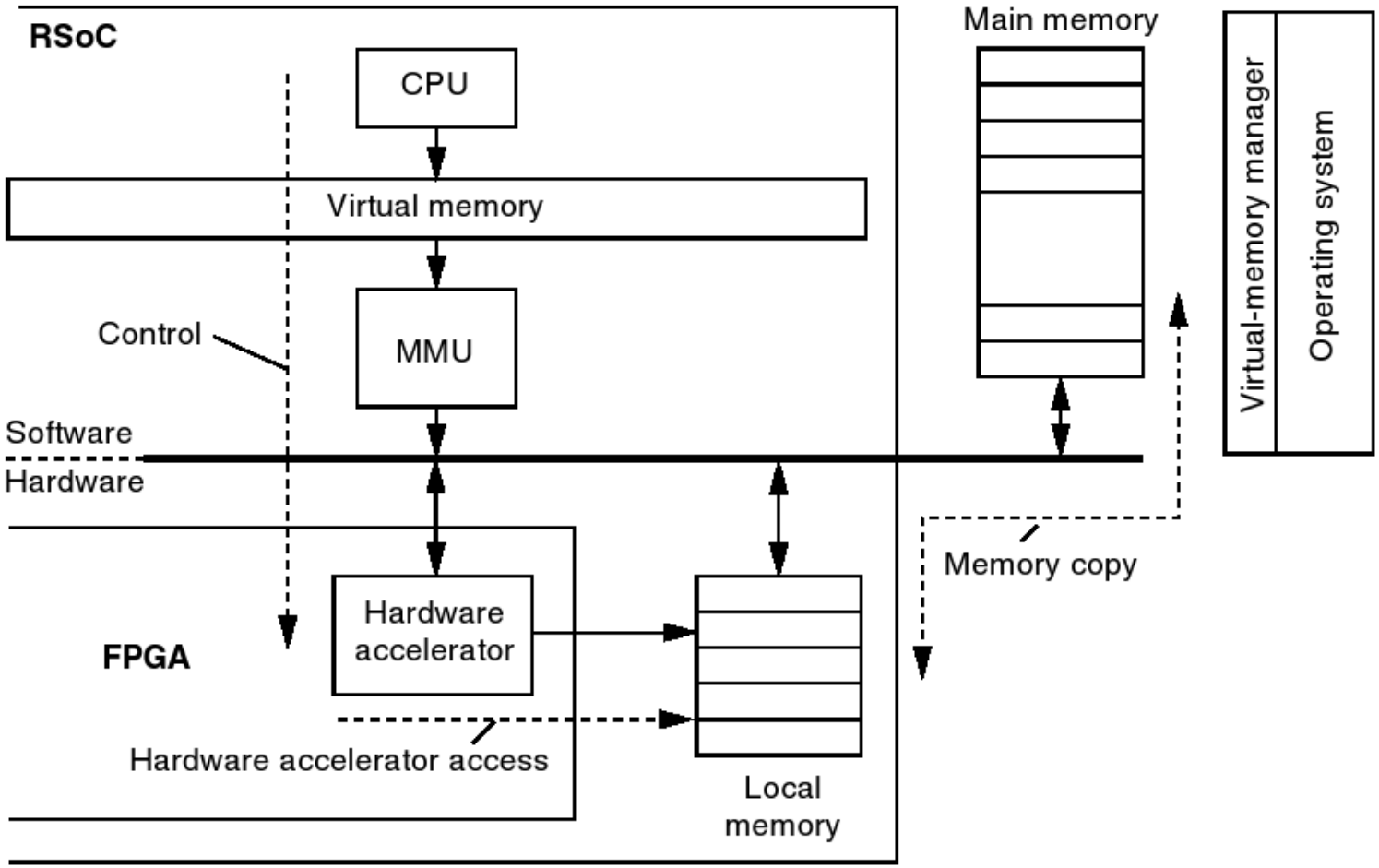
Kommunikationsinfrastruktur



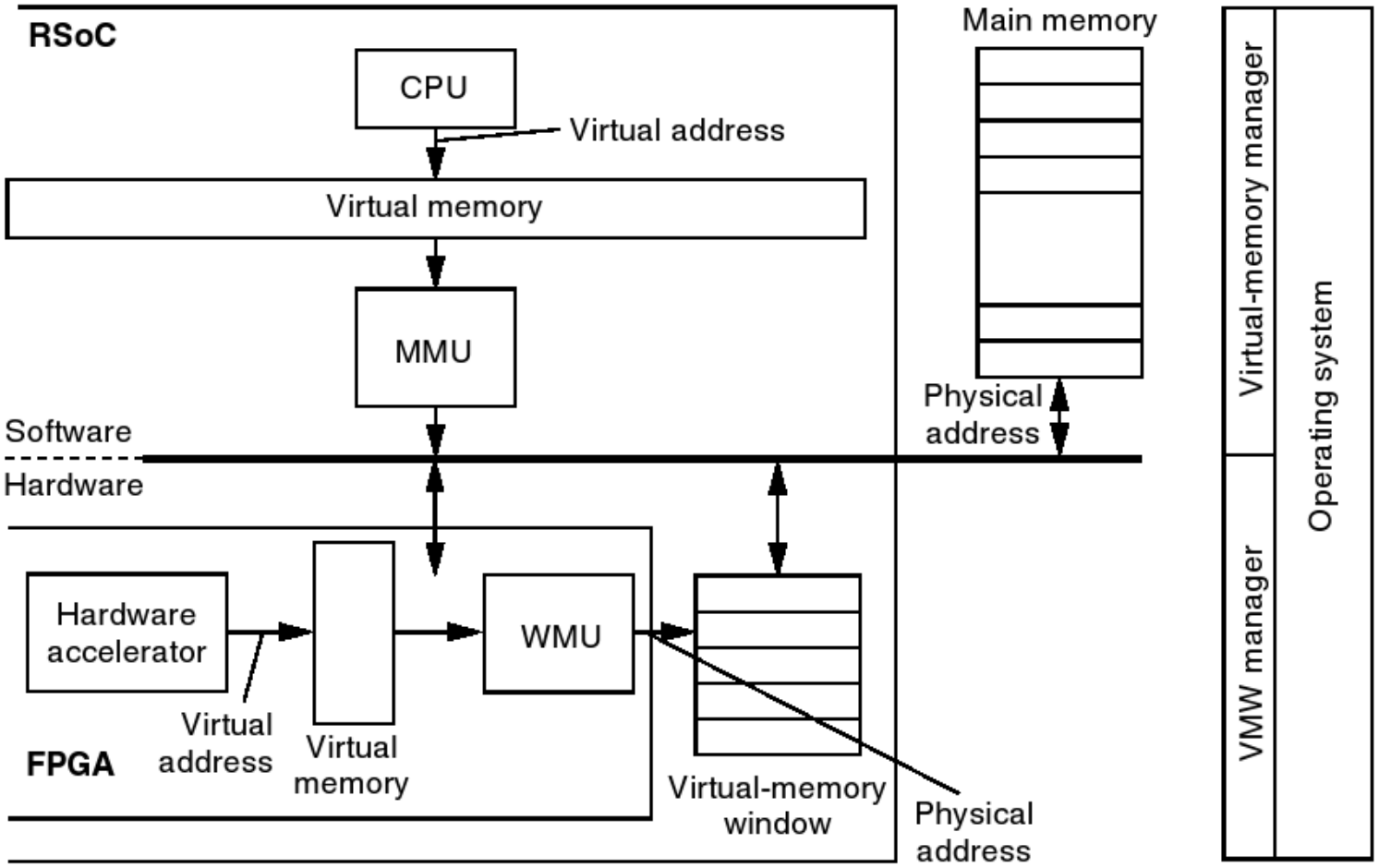
Pipelined

- hohe Parallelität
- relativ flexibel
- dynamisches Datenflussrouting erforderlich

HW Memory Management



HW Memory Management with Virtualization

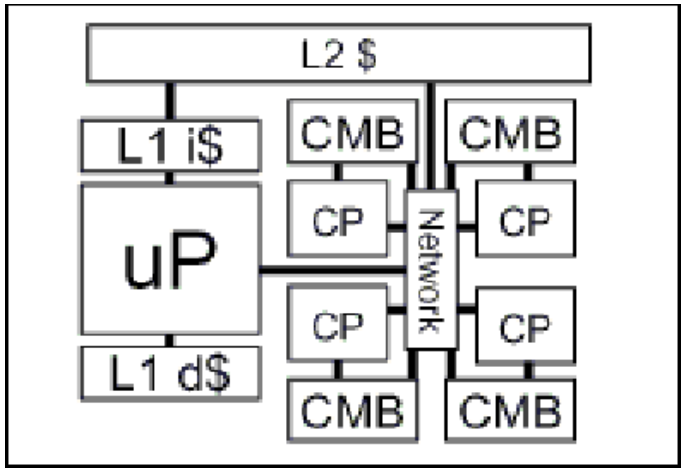


Beispiele für HOS

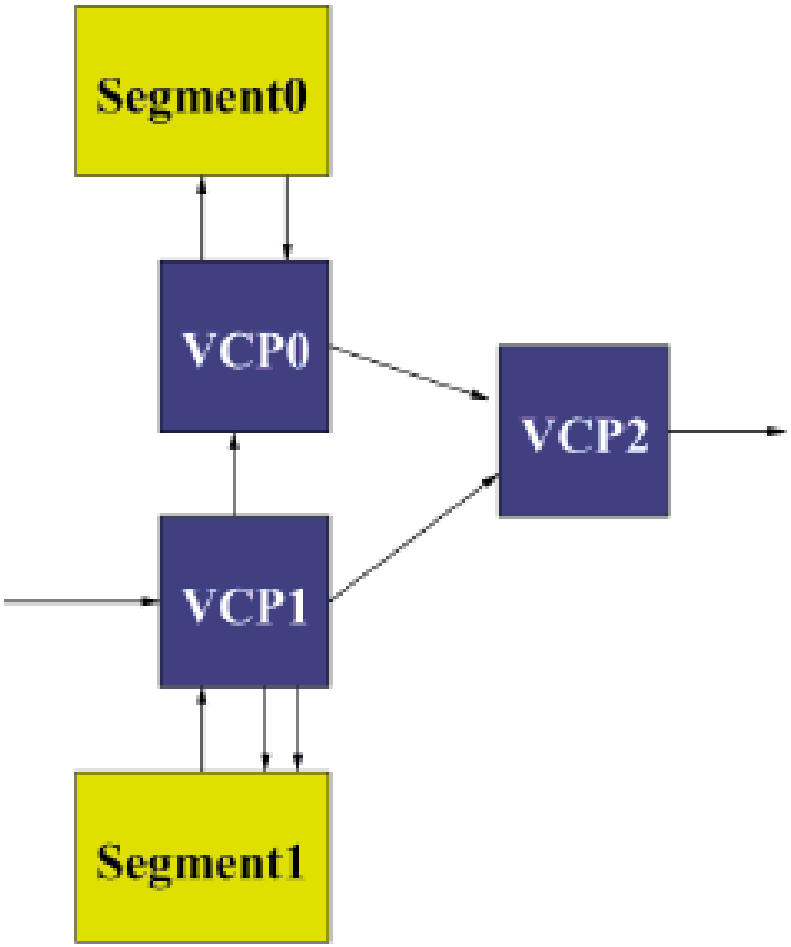
- → SCORE (Stream Computations Organized for Reconfigurable Execution), U Berkeley
- → Prototype OS, ETH Zürich
- → HTHREAD (hybrid thread), U Kansas
- ARCS (ein NoC), U Rostock
- DISC (Dynamic instruction set computer), U Provo
- DynaCore, U Lübeck
- OS4RS (Operating System for Reconfigurable Systems), U Brüssel / U Leuven
- HOPES (Hardware Operating System), U Singapore
- ReConfigME, U Berkeley

SCORE

- Operatoren können rekonfiguriert werden (Compute Page, CP)
- formale Modellierung des Datenflusses für Planung
- Speicher wird virtualisiert
- könnte auch auf Clustern eingesetzt werden



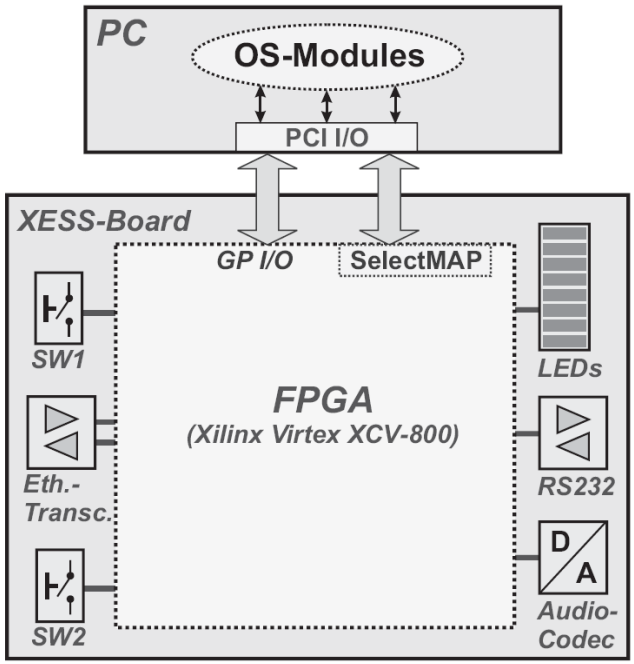
SCORE Programmiermodell



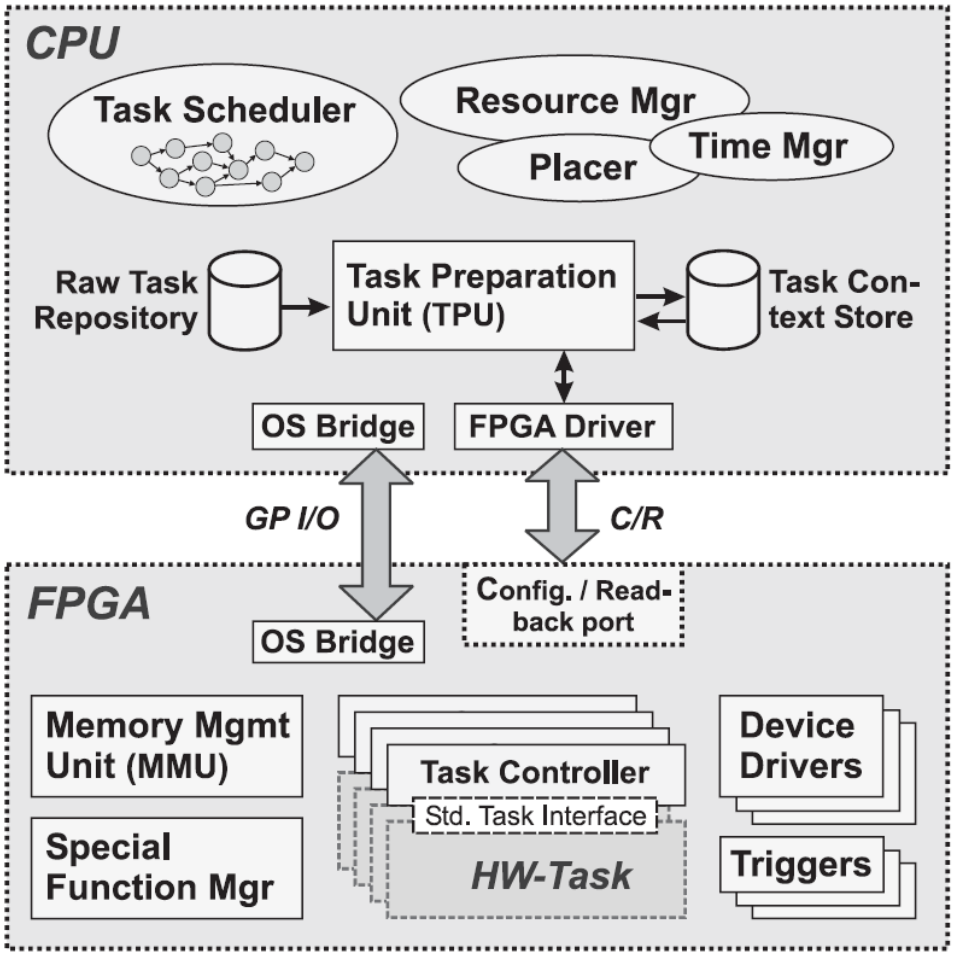
- Daten werden von Speicher zu Speicher bewegt
- Operanden bestimmen den Datenfluss
- Operanden werden exportiert und eingeplant

Prototype OS

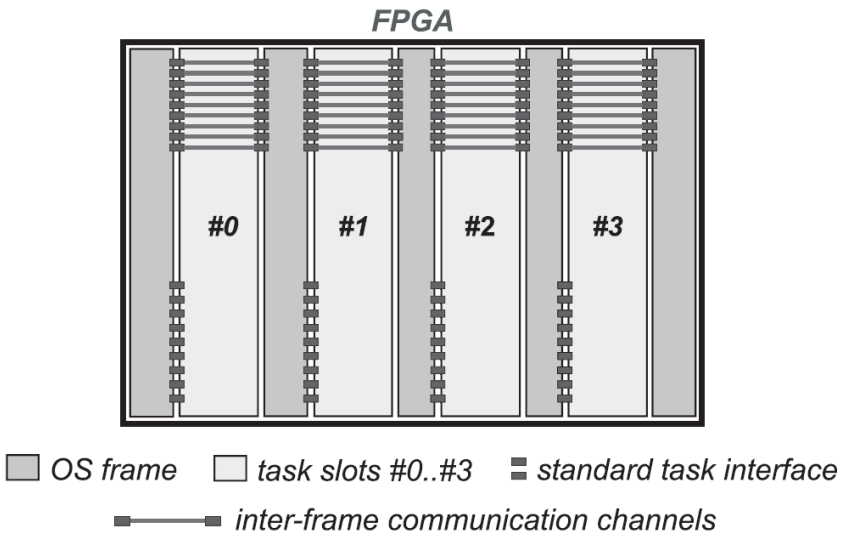
- OS und User (Task) Frames (FPGA)
- Hardware Tasks sind definiert durch
  - geometrische Größe
  - clock range
- Realisierung als PCI Karte



Architektur



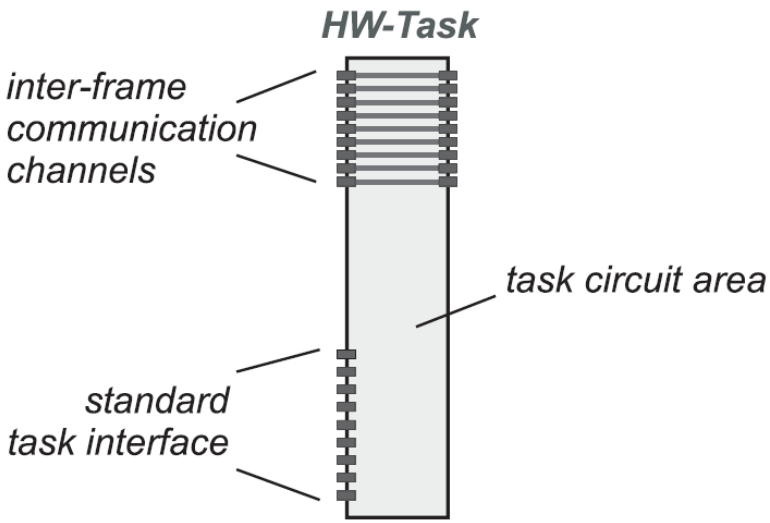
Task Interfaces (1/2)



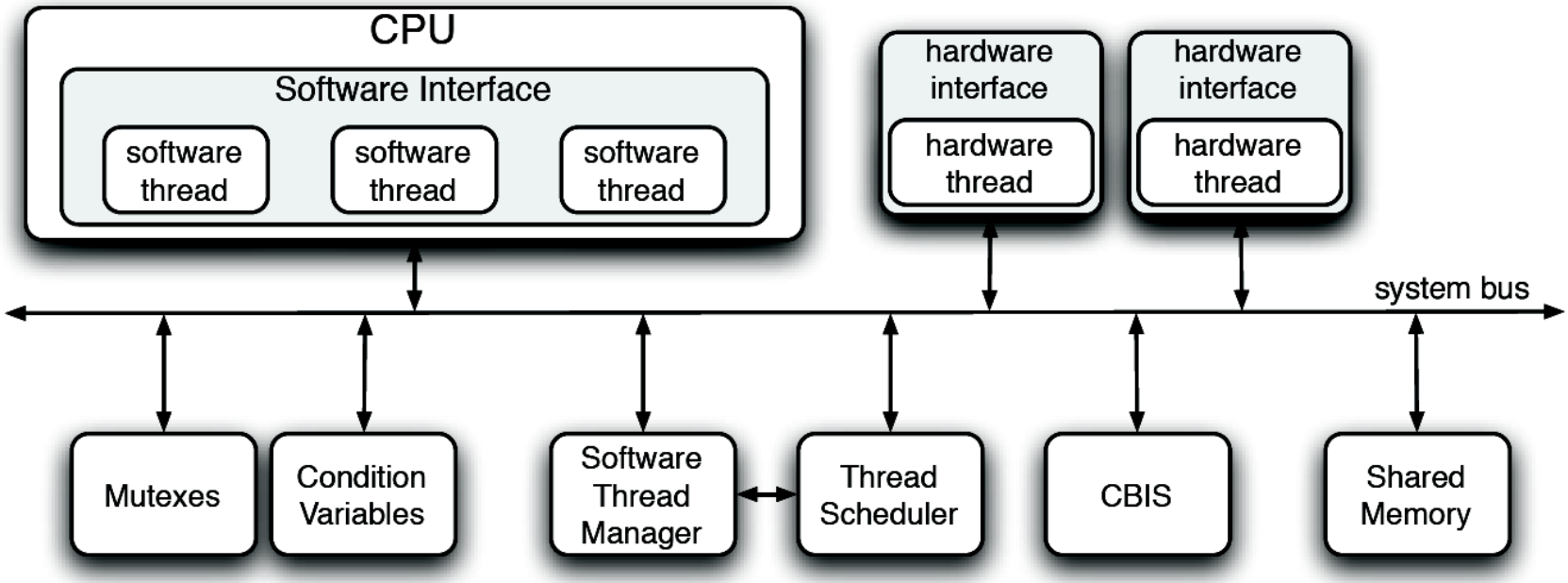
- Tasks reduziert auf Spalten
  $\longrightarrow$ Technologieabhängigkeit
- benutzt Xilinx Modular Design Package
- reduziert Routingproblem auf $O_{(1)}$
- reduziert Platzierproblem auf $O_{(n)}$

Task Interfaces (2/2)

- alle Tasks gleich groß
- jeder Task hat die gleichen Schnittstellen (Modular Design Package)
  - standard task interface (STI)
  - inter-frame communication channels (IFCC)

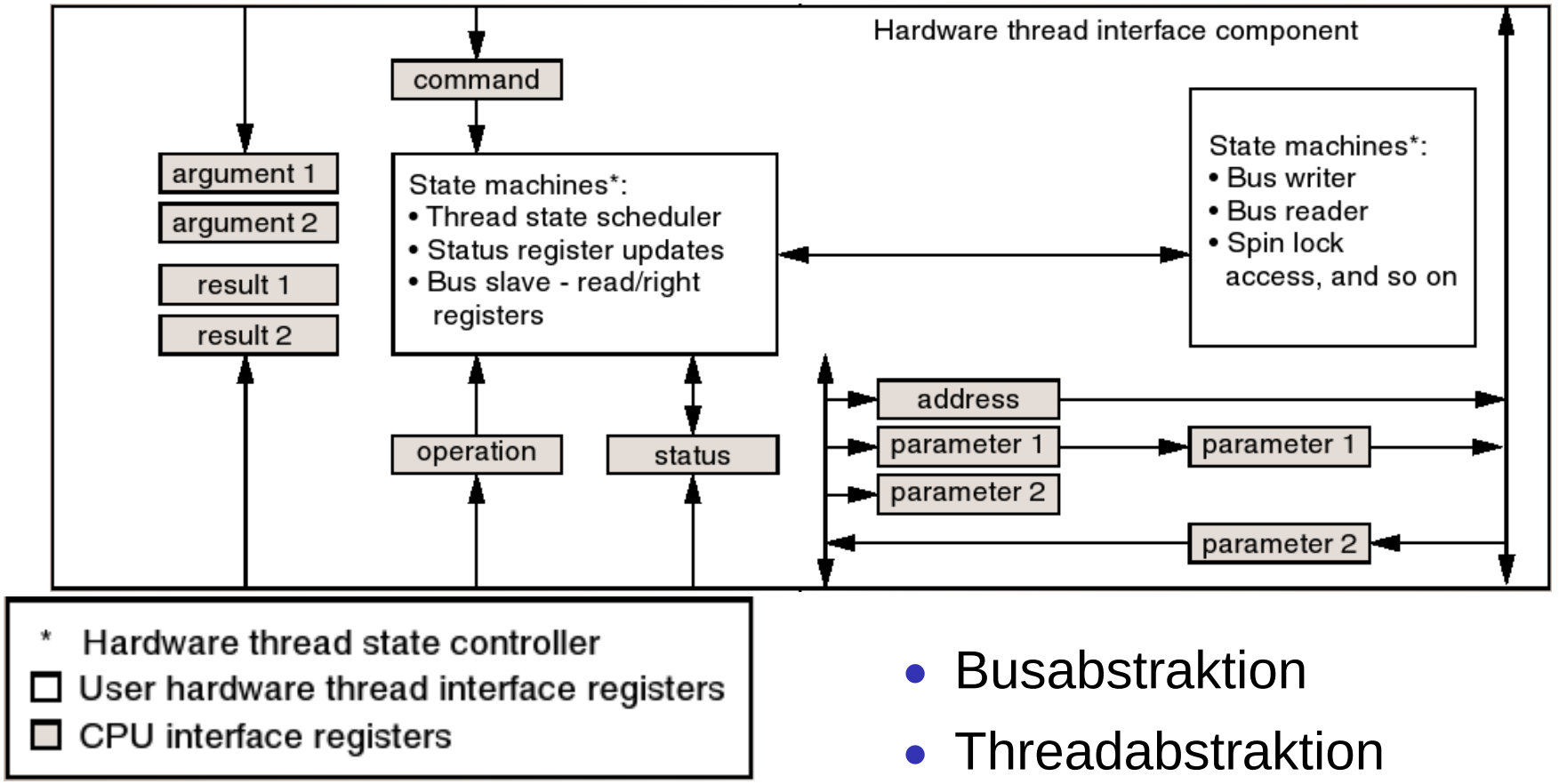


Hybridthreads - Aufbau



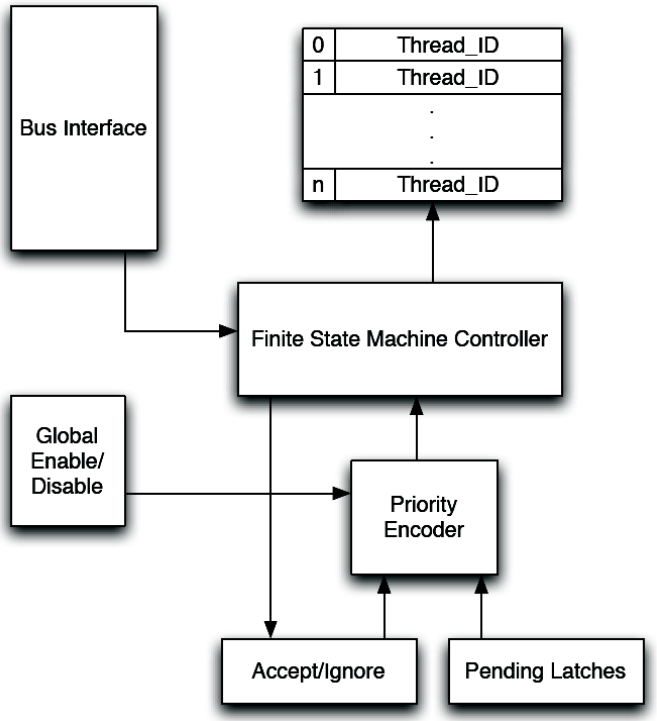
- Vereinheitlichung von Hard- und Softwarethreads
- POSIX kompatibles, hybrides Betriebssystem
- Programmiersprachen GIMPLE, C, VHDL

Hybridthreads - HWTI



- Busabstraktion
- Threadabstraktion

Hybridthreads CBIS



- CPU Bypass Interrupt Scheduler
- umgeht Unterbrechung der CPU
- löst Interruptpriorität über Threadpriorität der ISR
- Interrupt Delay 760 - 1410 ns (clock cycle 10ns)

Hybridthreads - Performance

	2 Running Software Threads			250 Running Software Threads		
	Min (µs)	Mean (µs)	Max (µs)	Min (µs)	Mean (µs)	Max (µs)
Scheduling Decision	1.750	1.751	2.140	1.910	1.975	3.380
Mutex Lock	.750	.750	.750	.750	.750	.750
Interrupt Handler Determination	.760	.760	.760	.760	.796	1.530

- kleiner Jitter
- linearer oder konstanter Zeitaufwand
- parallele Verarbeitung / Verwaltung

Wandel der Sichtweisen

Die Anfänge

- Wie kann man FPGAs rekonfigurieren?
- Wie nutzt man das?
- Was bedeuted das?
- kennenlernen der Materie

erste OSs / Simulationen

- Parititioning
- Scheduling (Unterbrechung?)
- die ersten HOSs wurden erstellt
- Software rekonfiguriert nur

Wandel der Sichtweisen

praktikable Implementationen

- Optimierung der Algorithmen
- erste funktionierende Systeme
- Hardware als Dienste für die Software

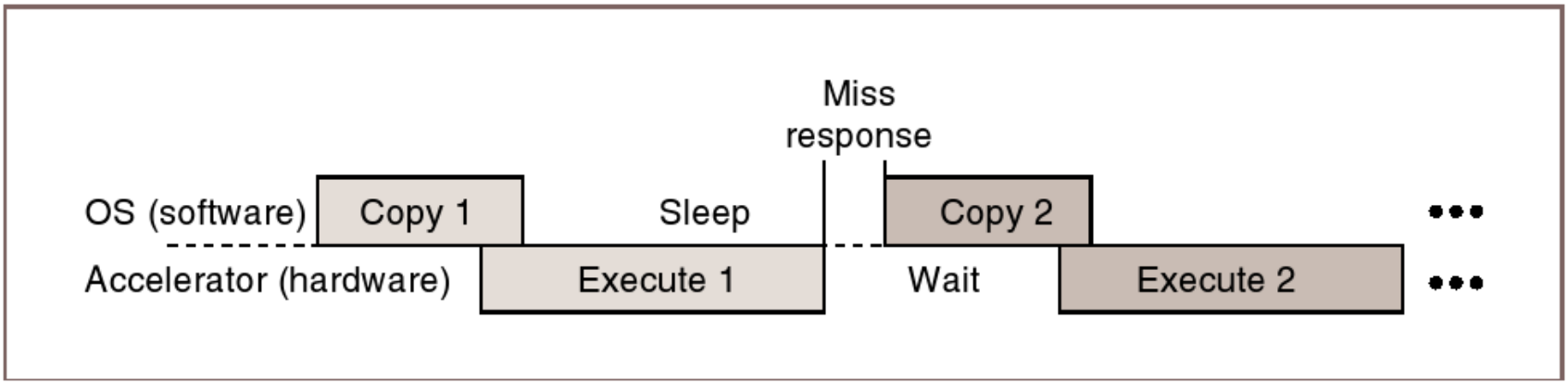
größere Systemsicht

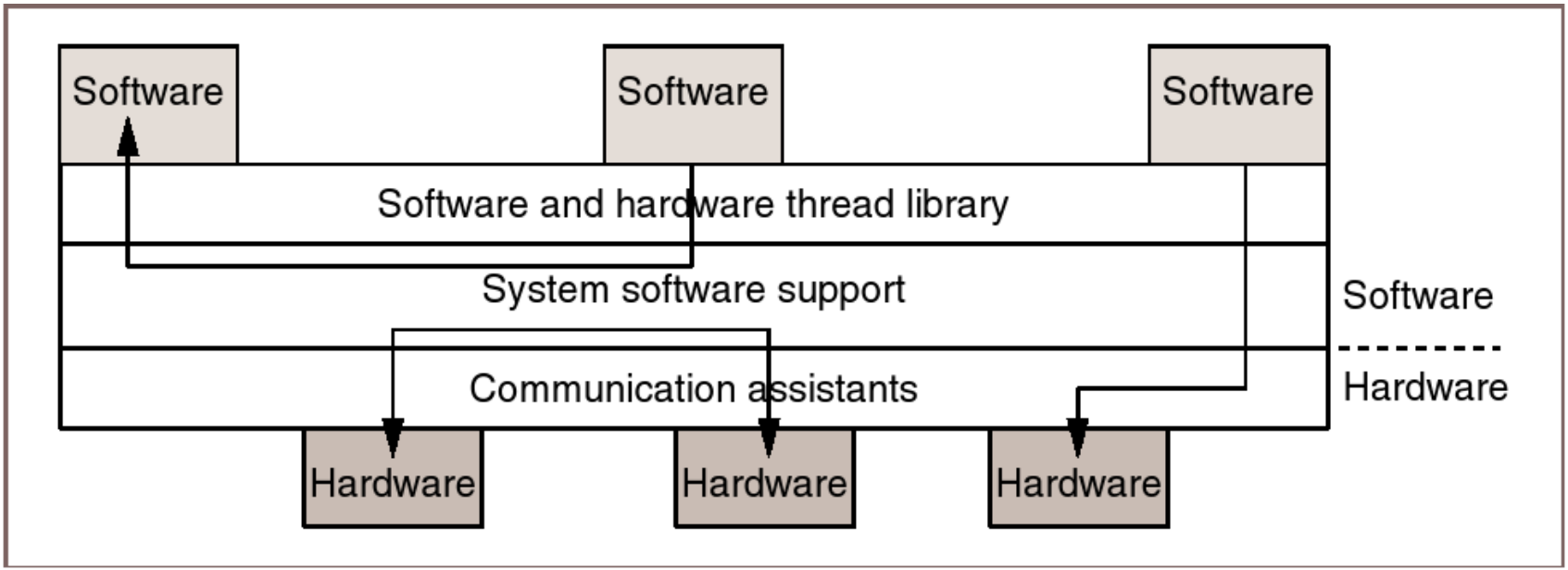
- höhere Abstraktion (Formalisierung?)
- Hardware und Software werden gleichberechtigt gehandhabt
- homogenere Designansätze
- erste Industrieprojekte (wearable computing, avionics)

Zusammenfassung

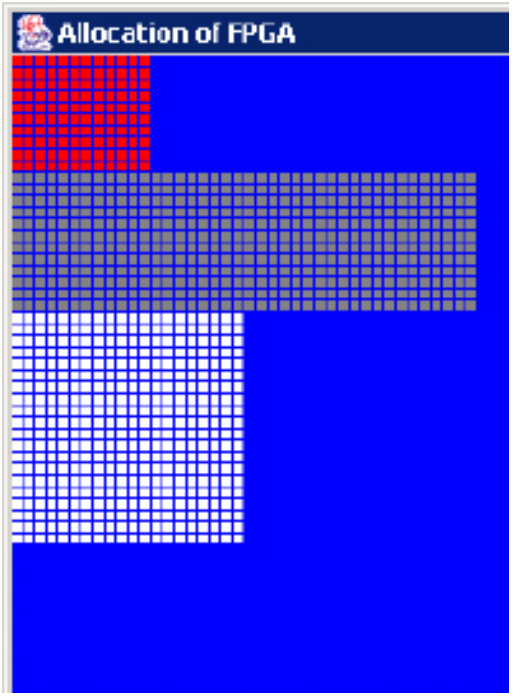
- Grundlagen wurden geschaffen (Algorithmen...)
- Fokussierung auf Effizienz und Machbarkeit
- Designflow wird optimiert
- Bedarf an höheren, allgemeineren Beschreibungssprachen HLL (High Level Languages)

Kommunikation im Shared Memory





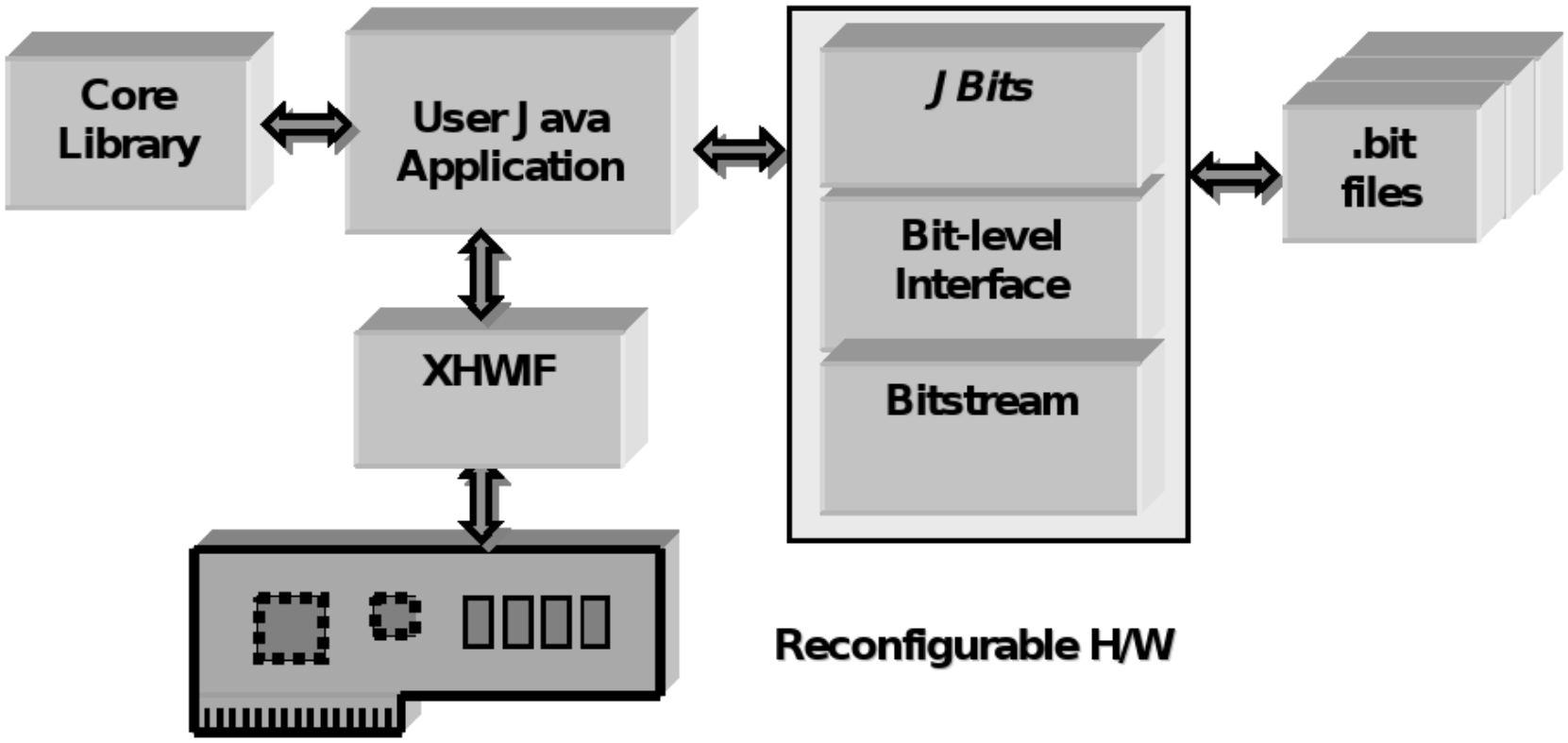
ReConfigME



- frühes System
- Java basiert
  Xilinx Jbits
- online Partitionierung / Routing
- Koprozessor als PCI Karte
- keine partielle Rekonfiguration möglich
  $\longrightarrow$ fehlende Technologie

Return

Xilinx Jbits



Return

verschiedene Design Flows

Getrennter Designfluss

- \+ spiegelt Entwicklerspezialisierung wieder
- \+ bessere Optimierung / Anpassung
- \- frühe Partitionierung - unflexibel

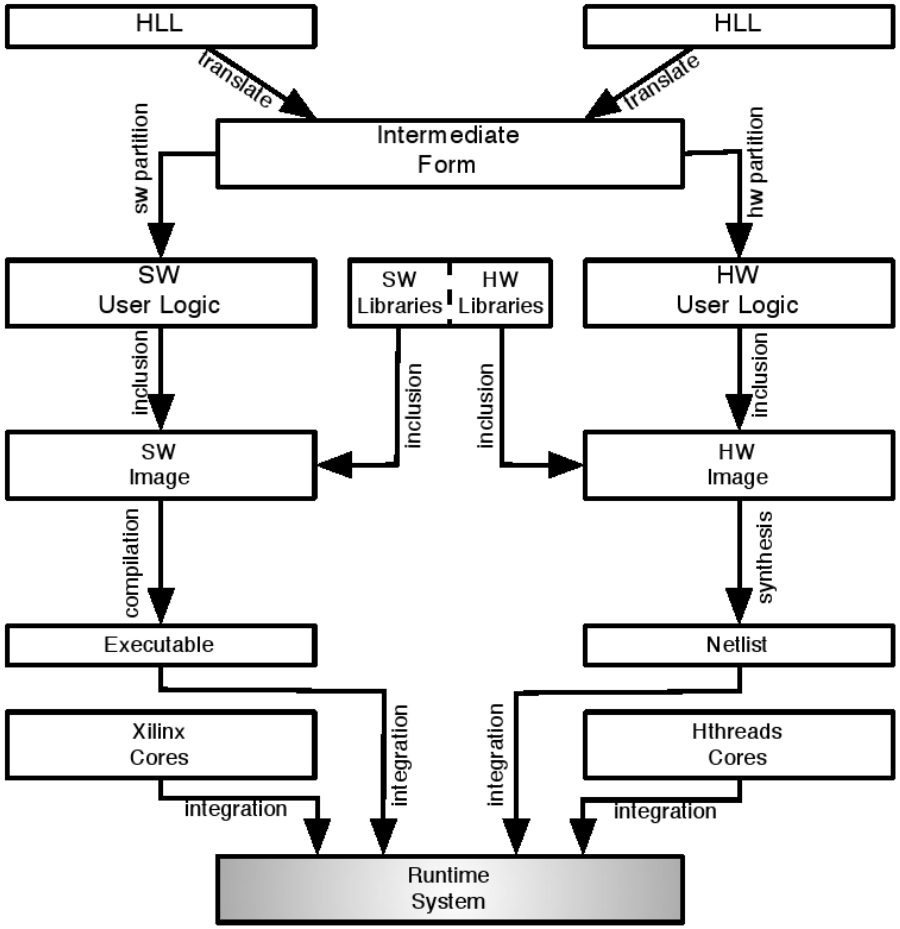
Komplettdesign in C

- \+ späte Partitionierung - flexibel
- \+ weit verbreitete Sprache
- \- gute C Synthesetools notwendig / bisher nicht optimierte HW

höhere Abstraktionssprache

- \+ späte Partitionierung - flexibel
- \+ relativ gut optimierte HW / SW
- \- unbekannte Sprache

HTHREAD Design Flow



Antworten zu Fragen aus dem vorherigen Vortrag

- Zeitpunkt der Rekonfiguration ist system- und anwendungsabhängig
- Welches Modul dann geladen wird, bestimmt im Idealfall auch nur die Anforderung zu dem Zeitpunkt
- Das Modul wird vom Loader über die Config- / Readbackport des FPGA geladen