FPGA Beschleuniger

Armin Jeyrani Mamegani

HAW Hamburg
Department Informatik

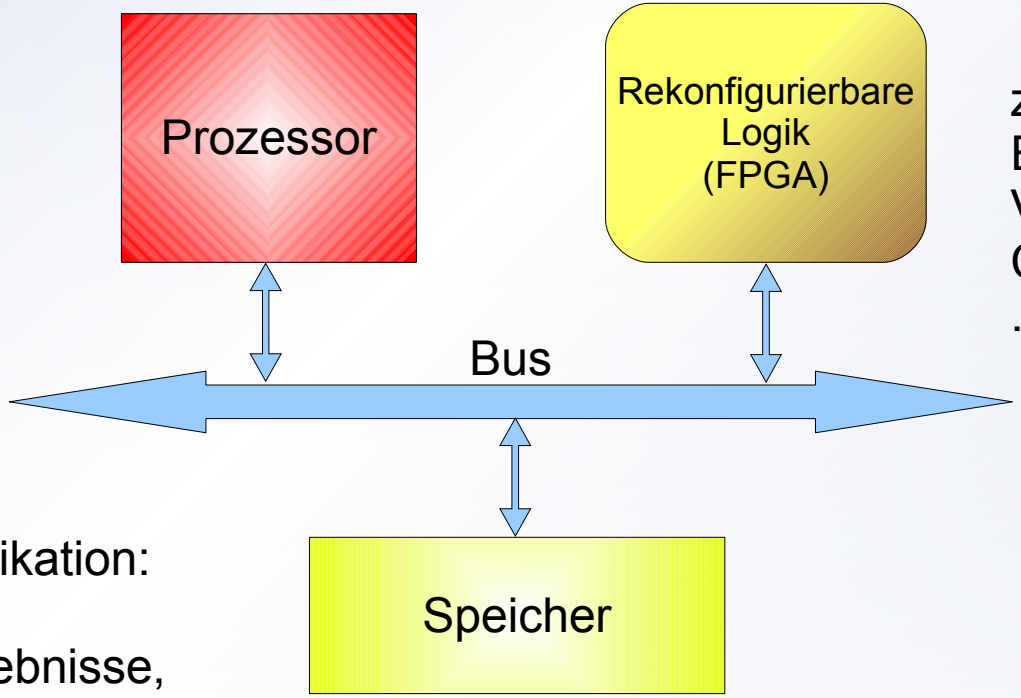
15.12.2008

Einleitung

- Wiederholung aus AW1: Handy Plattform, Dynamic Reconfigurable Computing

SW:
Kontrolle

z.B.:OS



HW:
Rechenintensive Tasks,

z.B.:
Encryption,
Video Compression,
...

HW/SW Kommunikation:
Parameter,
Zwischenergebnisse,
...

Einleitung

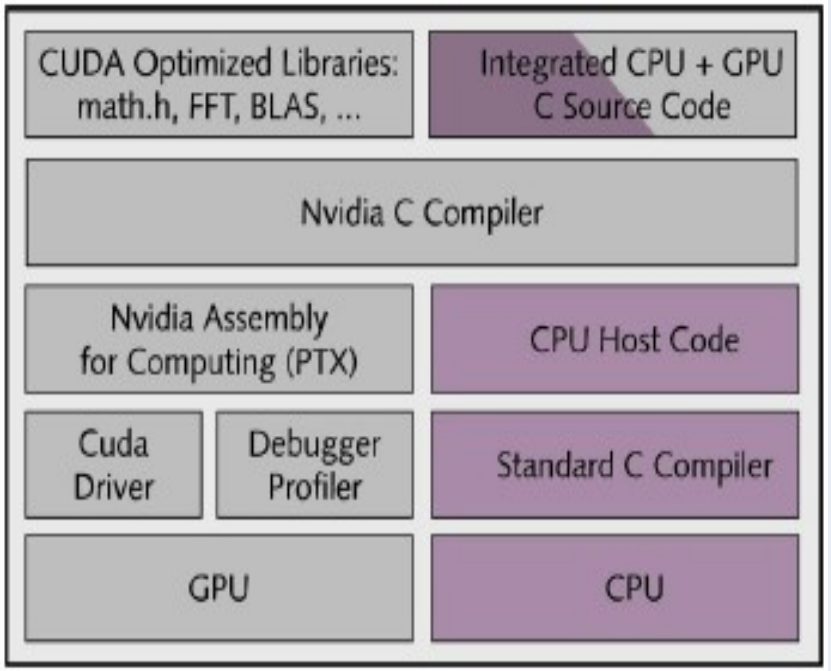
- Bessere Idee
  - Warum Handy Plattform?
  - Warum nicht allgemeine Plattform?
    - z.B. für PC, Handy, Embedded Computer,...
- Voraussetzung
  - Beschriebene Architektur (Prozessor, FPGA, Speicher, Bussystem)
- Problem: Entwickler
  - Kennt die HW nicht und will die HW auch nicht kennen
  - Zu wenig Entwickler, die sowohl SW (z.B. C) als auch HW (z.B. VHDL) entwickeln können
  - Aber C kann fast jeder

Einleitung

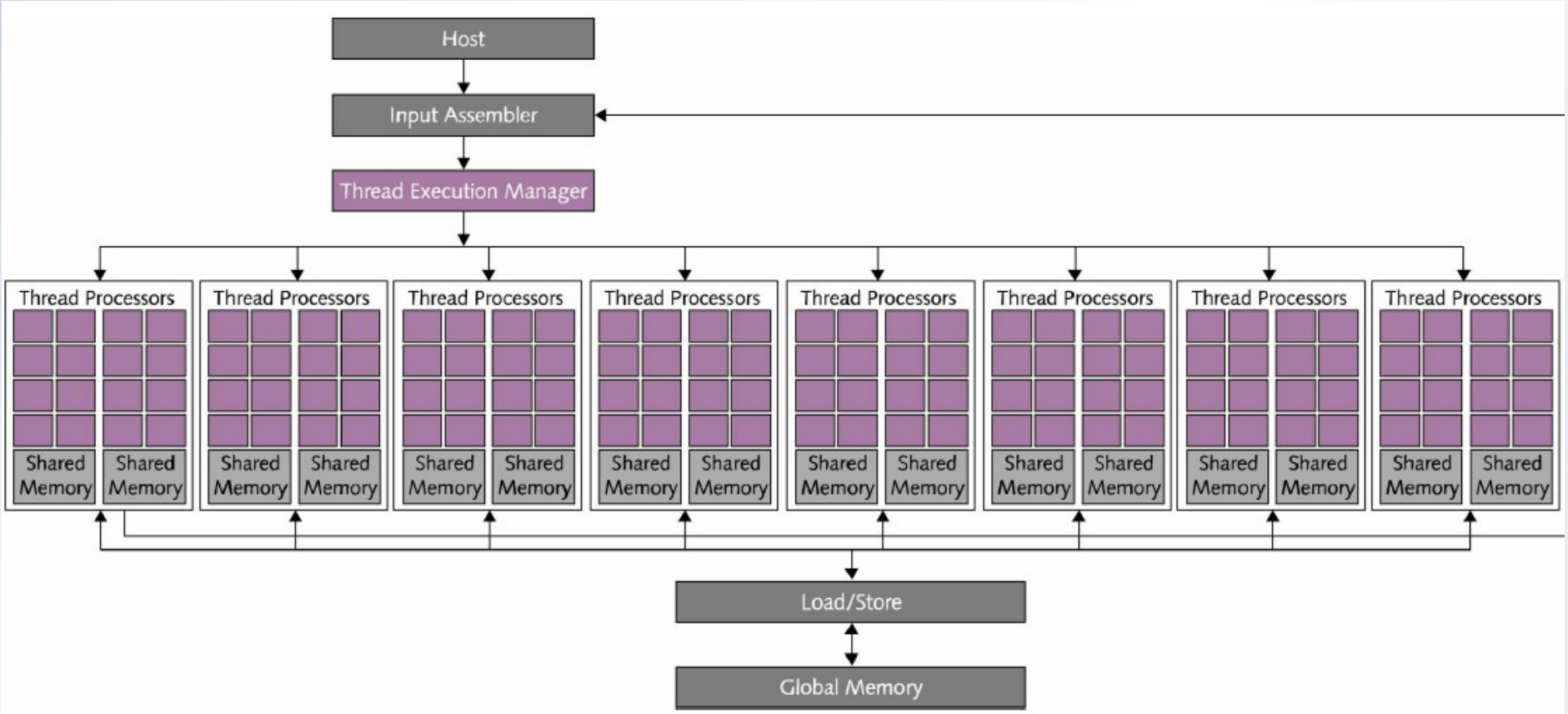
- Entwicklung im PC Segment
  - Auslagerung rechenintensiver Berechnungen auf den Grafikprozessor
  - NVIDIA CUDA, AMD Stream Computing, OpenCL
- Entwicklung im Server Segment
  - Zum Teil Grafikprozessorsysteme (NVIDIA)
  - Zusätzlicher FPGA zu AMD Opteron Prozessoren
- FPGA-Beschleuniger gibt es im PC-Segment nicht
  - Was machen NVIDIA und Co?

NVIDIA CUDA

- Compute Unified Device Architecture
- Platform für paralleles high-performance computing auf der GPU
- C/C++ Entwicklungswerkzeuge
- GPU-HW transparent für den Entwickler
- Threadmgmt. transparent für den Entwickler
- Geeignet für datenintensive Applikationen mit einfach-genauen Fließkommaberechnungen



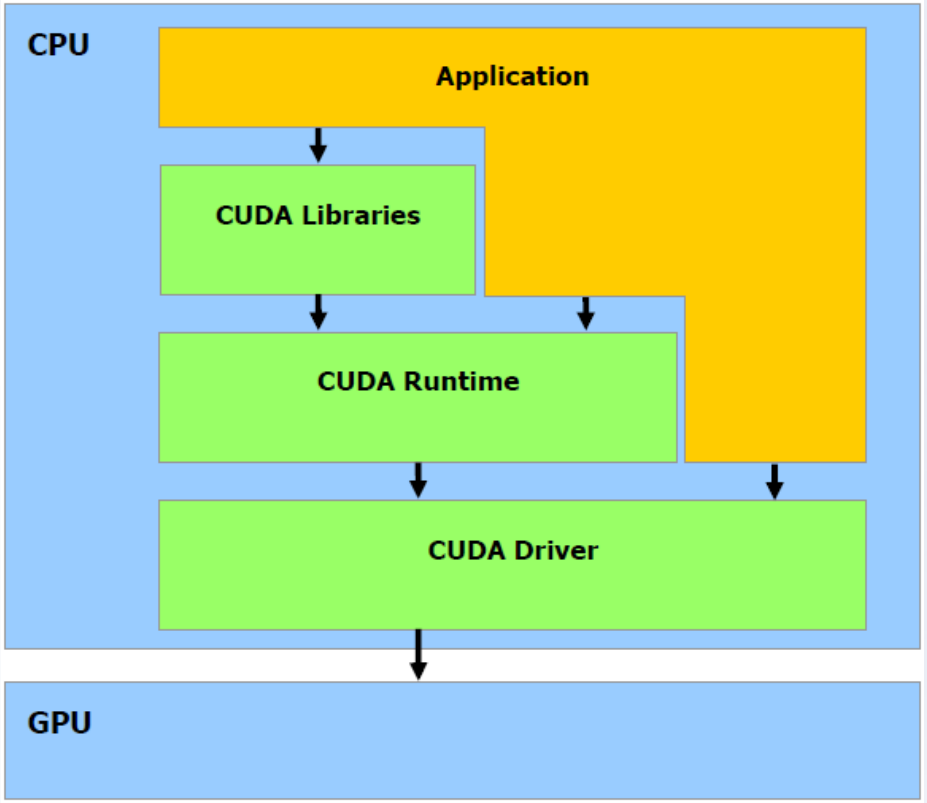
NVIDIA CUDA



- Geforce 8 GPU Architektur
  - 128 Thread-Prozessoren
  - HW Thread Execution Manager
  - SIMD Architektur

NVIDIA CUDA

- Software-Stack
- CUDA Libs
  - FFT, BLAS
- CUDA Runtime API
  - Hoch abstrakte Funktionen
- CUDA Driver API
  - Besserer Zugriff auf GPU
- GPU-Code wird in C Syntax geschrieben
- Cuda Parser teilt den Code in GPU und CPU Code auf (__global__ für GPU Code)



NVIDIA CUDA

- Pro
  - Beschleunigt rechenintensive Applikationen
  - Frei erhältliches SDK
  - NVIDIA Grafikkarten weit verbreitet
  - Kompatibilität zu zukünftigen GPUs sichergestellt
- Kontra
  - Fest verdrahtet
    - Flexibilität liegt nur in der Software
  - Abhängig von der Produktlinie
  - Abhängig vom Hersteller
  - Grafikprozessor auch manchmal mit Grafikberechnung beschäftigt
- AMD ähnlich

OpenCL

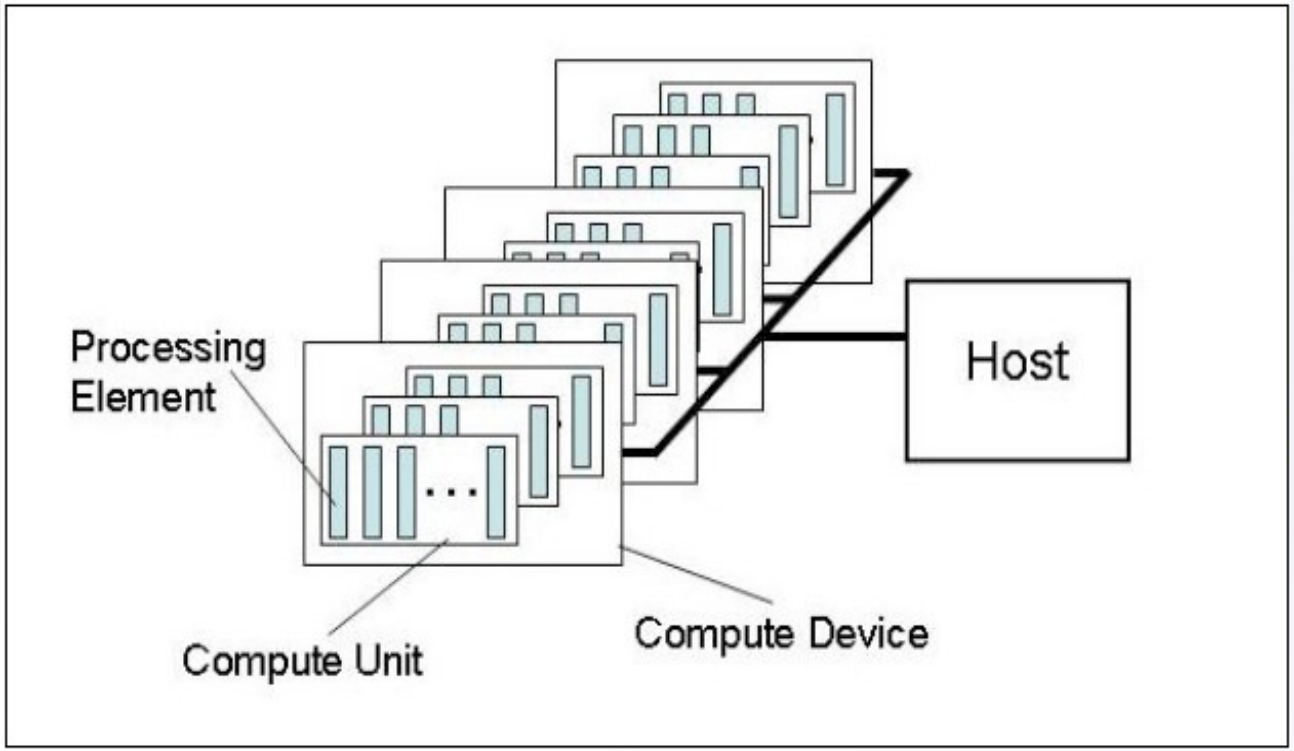
- Open Computing Language
- Nutzung aller Recheneinheiten eines Systems
  - CPUs, GPUs, DSPs
- Erlaubt daten- und taskparalleles Computing
- C-Basiert (ISO C99) mit speziellen Erweiterungen für paralleles Programmieren
- Funktioniert so aber nicht mit FPGAs
- Einfacher Treiber für FPGA nicht ausreichend
- Synthesefähige (Zwischen-)Sprache wird benötigt
- Kommunikationsmodell wird benötigt

OpenCL

- Platform API
  - Abstraktion über diverse Devices
  - Devices zur Laufzeit entdecken, auswählen und initialisieren
  - Rechenrahmen und Ausführungsqueues erstellen
- Runtime API
  - Ausführen von Kernels
  - Mgmt. vom Scheduling und Memory
- Mehrere Sichten
  - Platform Model
  - Execution Model
  - Memory Model
  - Programming Model

OpenCL

- Platform Model
- Strukturiert in
  - Compute Device
    - Compute Unit
      - Processing Element



OpenCL

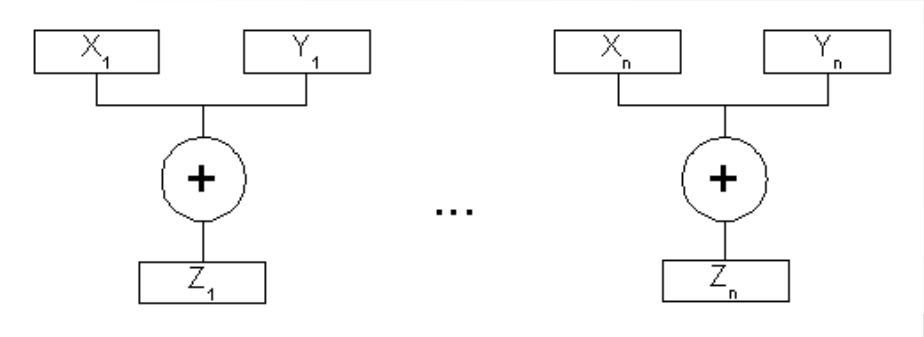
- Execution Model
  - Compute Kernel
    - Basisseinheit, vergleichbar mit C-Funktion
    - daten- oder taskparallel
    - Kernelinstanzen werden auf den Devices ausgeführt
  - Host-Programm
    - Kontextdefiniton und Ausführungsmgmt. für die Kernel
  - Kontext
    - Devices
    - Kernels: OpenCL Funktionen, die auf den Devices ausgeführt werden
    - Program Objects: Kernelcode und Executable
    - Memory Objects: Speicher, sichtbar für Host und Kernel
  - Command-Queue: Koordiniert die Ausführung der Kernel auf dem Device

FPGA Beschleuniger

- Beispiel: Vektoraddition
  - $x_1,...,x_n + y_1,...,y_n$
  - Code:

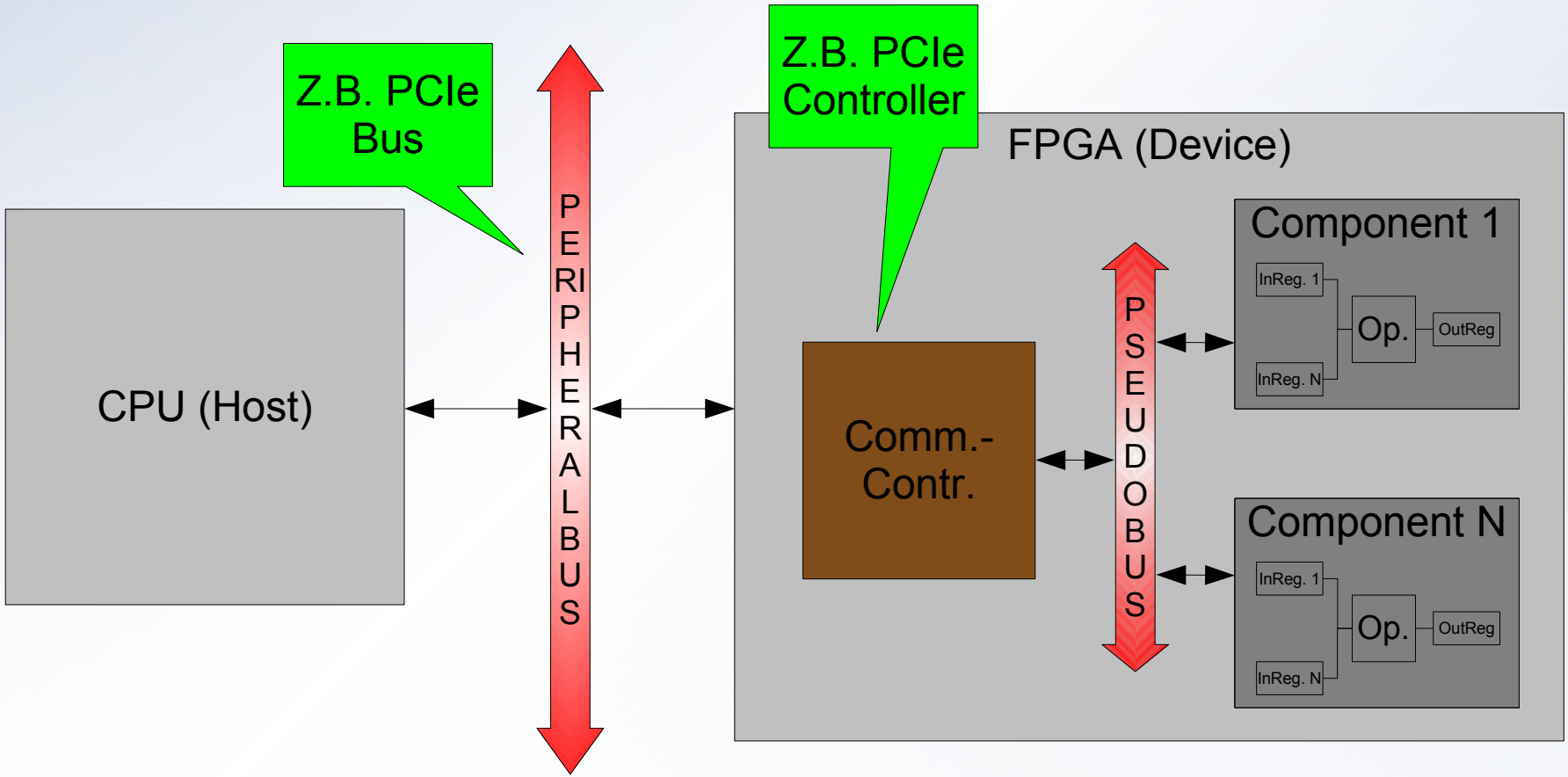
```
for (i = 0; i < n; i++)
        z[i] = x[i] + y[i];
```

  - CPU: sequentiell
    - n x Ausführungszeit
  - FPGA: parallel (datenparallel)
    - 1 x Ausführungszeit



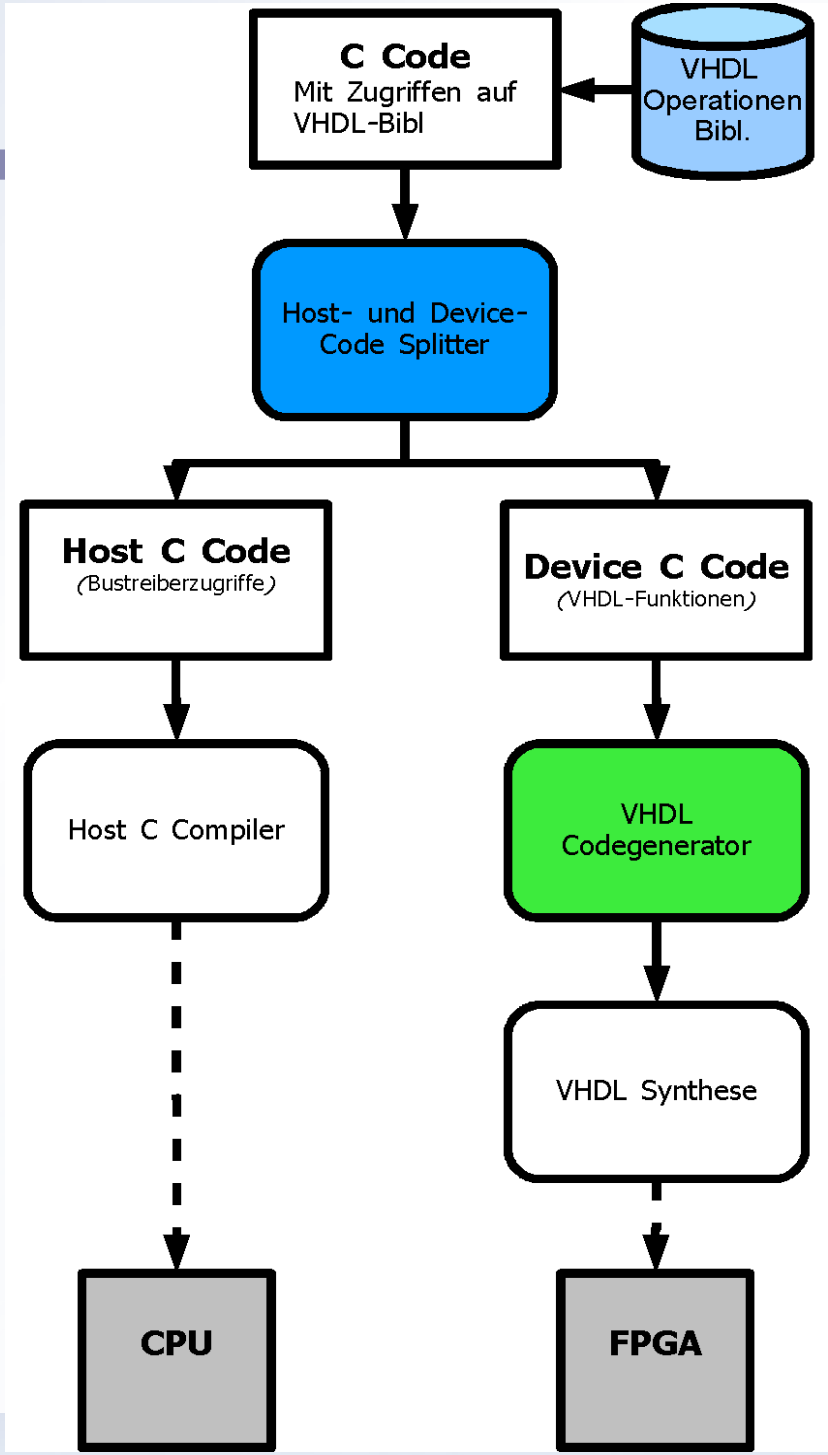
FPGA Beschleuniger

- HW-Aufbau
  - Operationen werden als Komponenten im FPGA realisiert



FPGA Beschleuniger

- Framework
  - VHDL-Bibl.
  - Code-Splitter
  - Code-Generator
  - Ausführungsskript
  - FPGA Runtime
    - FPGA Treiber
    - Laufzeitkonfiguration
    - FPGA Mgmt.
    - Zugriff auf FPGA



FPGA Beschleuniger

- Zur Laufzeit
  - Applikation wird gestartet
  - FPGA wird automatisch zur Laufzeit Programmiert
  - Sobald FPGA bereit, erhält es die Daten zum Rechnen
  - Kommunikation zw. Hostprogramm und FPGA erfolgt durch Lesen und Beschreiben von Registern
  - Applikation wird beendet
  - FPGA steht anderen Applikationen zur Verfügung
- FPGA Runtime muss diese Abstraktion bereitstellen

Ausblick

1. Codeumsetzer
   - Code für den Zugriff auf den Bus und an die richtigen Register muss in den Code eingefügt werden
2. Bibl. für einfache VHDL Operationen
   - z.B.: unsigned char fpga_add8(unsigned char a, ...)
3. Einsatz des Codeumsetzers anhand einer Beispielappl.
4. FPGA Runtime
5. Skript oder Programm für die Automatische Abfolge
6. Evaluierung durch Aufzeigen eines Performancegewinns

Literatur

Wayne Wolf
High Performance Embedded Computing, Elsevier(MK), 1. Auflage, 2007

Brian Bailey, Grant Martin and Andrew Piziali
ESL Design and Verification. Morgan Kaufmann, 2007

Scott Hauck and Andre DeHon
Reconfigurable Computing. Morgan Kaufmann, 2008

AMD
AMD Stream Computing, User Guide, Oktober 2008

Tom R. Halfhill
Parallel Processing with CUDA, Microprocessor Report, Januar 2008

NVIDIA
NVIDIA CUDA, Refenrence Manual, Version 2.0, Juni 2008

NVIDIA
NVIDIA CUDA, Programming Guide, Version 1.1, November 2007

Govindaraju et. al.
High Performance Discrete Fourier Transforms on Graphics Processors, Microsoft Corp., November 2008

Aaftab Munshi
The OpenCL Specification, The OpenCL Working Group, Version 1.0, Dezember 2008

Neil Trevett
OpenCL, The Open Standard for Heterogenous Parallel Programming, November 2008

Aaftab Munshi
OpenCL, Parallel Computing on the GPU and CPU, SIGGRAPH 2008