Entwurf eines FPGA-Cores zur Simulationsbeschleunigung zeitkontinuierlicher Modelle im HiL Kontext

Till Fischer

03.11.2011

FZI Forschungszentrum Informatik
Embedded Systems & Sensors Engineering (ESS)

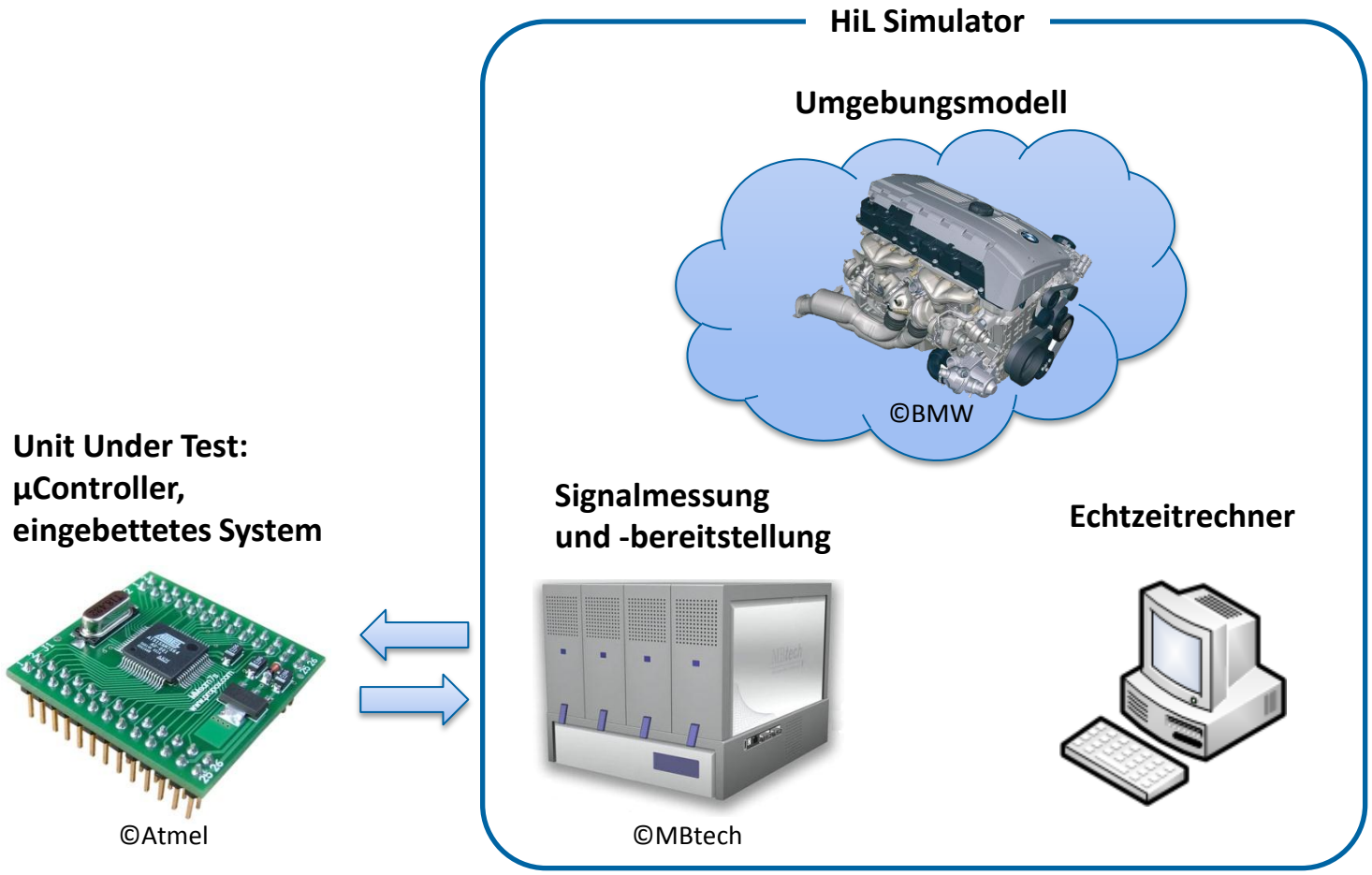
Diplomarbeit FZI / KIT (ehem. Universität Karlsruhe), Februar 2011

Überblick

- Motivation
- Lösungsansatz
- Architektur
- Ergebnisse
- Ausblick

Motivation

- Hardware-in-the-Loop (HiL) Test
  - Umgebungssimulation für Steuergeräte z.B. im Automobilbereich



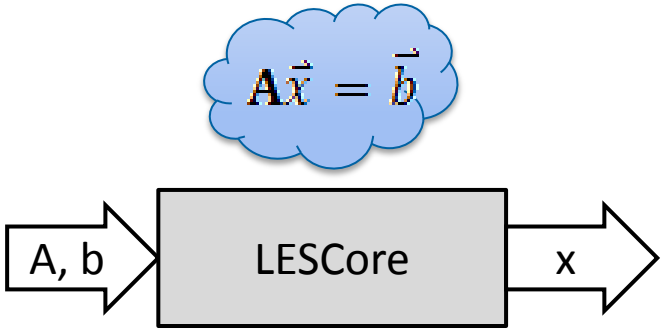
Motivation

- Beschreibung der Umgebung mittels Differentialgleichungen
- Sehr kurze Zykluszeit: bis in den Bereich von µs (!)
- Problem wenn DGL nicht im Voraus lösbar
  - neue Lösung in jedem Iterationsschritt erforderlich
  - Echtzeitanforderung

- Verwendung numerischer Verfahren
- Explizite und implizite Verfahren
  - Implizite Verfahren: erhöhte numerische Stabilität
  - Lösung von Gleichungssystem in jedem Iterationsschritt

LESCore: IP-Core für FPGAs

- LESCore: Linear Equation Solver Core
- IP-Core zur Lösung linearer Gleichungssysteme in Hardware



Lösungsansatz

Direkte Verfahren	
Gauß-Algorithmus	Umformung Koeffizientenmatrix A in Rechtsdreiecksmatrix Lösen durch „Rückwärtseinsetzen“
LR-Zerlegung	Abwandlung des Gauß-Algorithmus: Zerlegung von A in zwei Matrizen L und R mit A=L·R Wiederverwendbarkeit der Zerlegung
QR-Zerlegung	Weniger anfällig für Rundungsfehler, doppelte Laufzeit
Cholesky-Verfahren	Nur für symmetrische, positiv definite Matrizen

Iterative Verfahren
z.B. Krylow-Unterraum Verfahren Effizient für sehr große Systeme Konvergenzverhalten abhängig von Initiallösung Mehr Kontrollfluss als direkte Verfahren

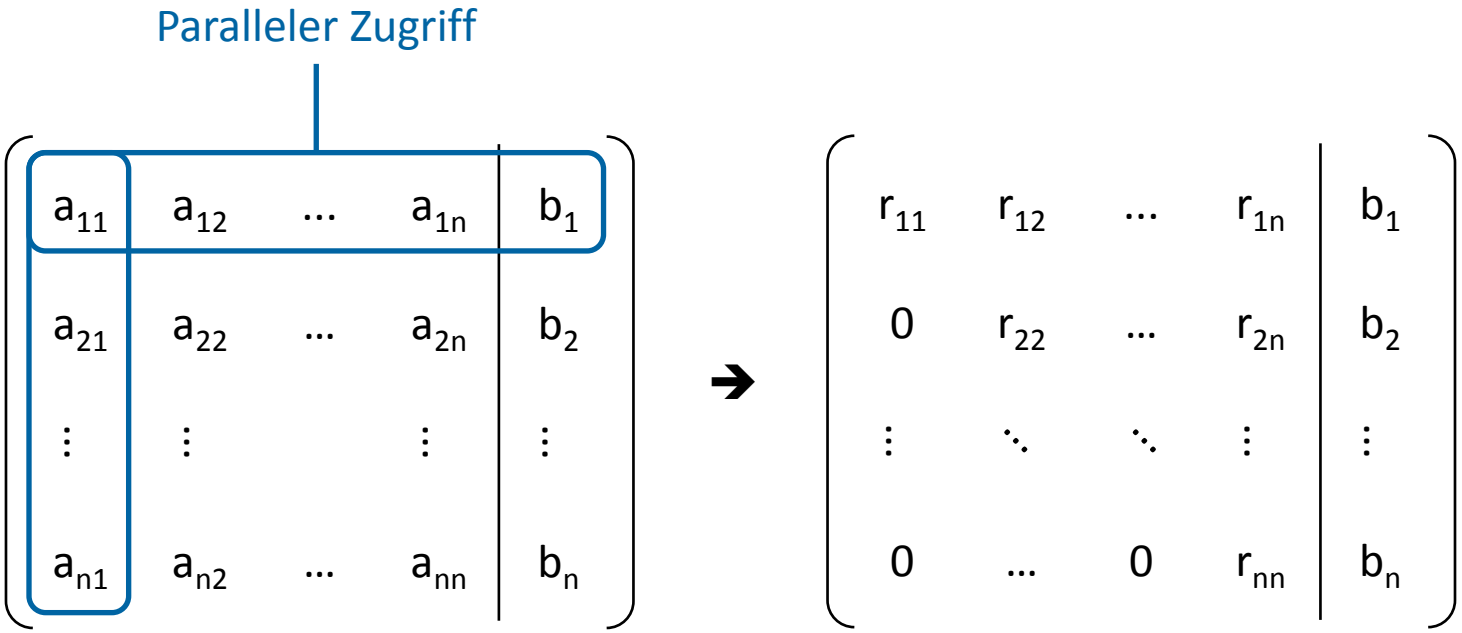
Lösungsansatz

Direkte Verfahren	
Gauß-Algorithmus	Umformung Koeffizientenmatrix A in Rechtsdreiecksmatrix Lösen durch „Rückwärtseinsetzen“
LR-Zerlegung	Abwandlung des Gauß-Algorithmus: Zerlegung von A in zwei Matrizen L und R mit A=L·R Wiederverwendbarkeit der Zerlegung
QR-Zerlegung	Weniger anfällig für Rundungsfehler, doppelte Laufzeit
Cholesky-Verfahren	Nur für symmetrische, positiv definite Matrizen

Iterative Verfahren

z.B. Krylow-Unterraum Verfahren
Effizient für sehr große Systeme
Konvergenzverhalten abhängig von Initiallösung
Mehr Kontrollfluss als direkte Verfahren

Lösungsansatz

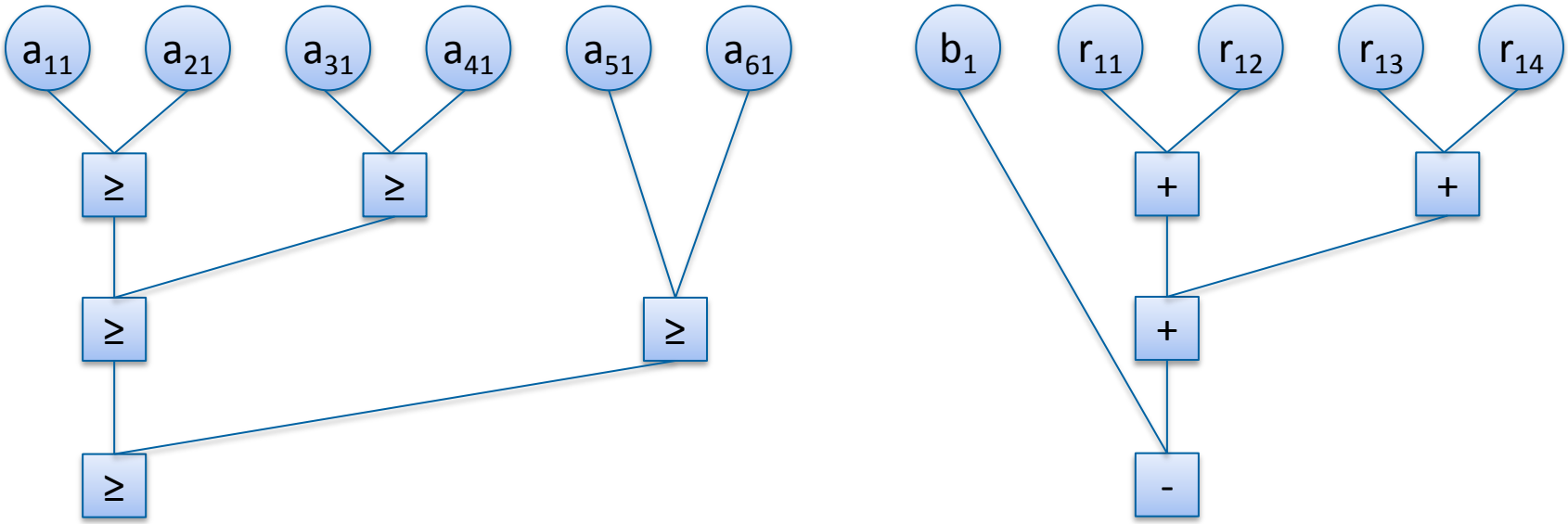


- Erzeugung von „Nullen“ unterhalb der Hauptdiagonalen durch Linearkombination von Zeilen
- Pivotierung/Zeilenvertauschung: Vermeidung von Rundungsfehlern
  - Suche nach Maximum innerhalb einer Spalte

- Breite Speicherschnittstelle für parallelen Zugriff
- Replizierte Recheneinheiten (SIMD)
- Cache für Koeffizienten einer Spalte
- Pivotierung löst außerdem Problem „Division durch Null“

Lösungsansatz

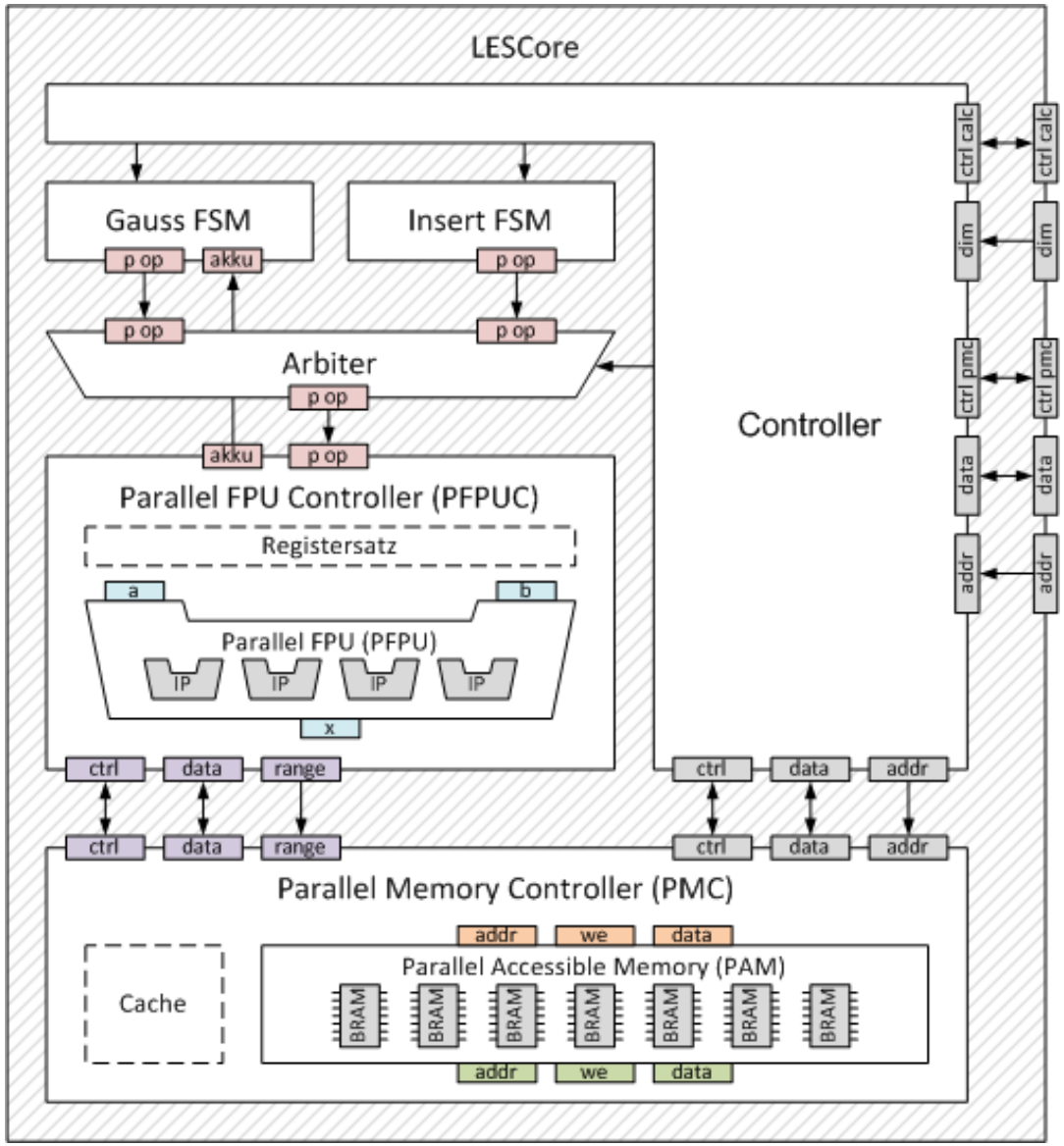
- Ausreizen der möglichen Parallelität
  - Suche nach Maximum (Pivotierung)
  - „Einsetzen“ zur Bestimmung des Lösungsvektors



LESCore Architektur

Designparameter:

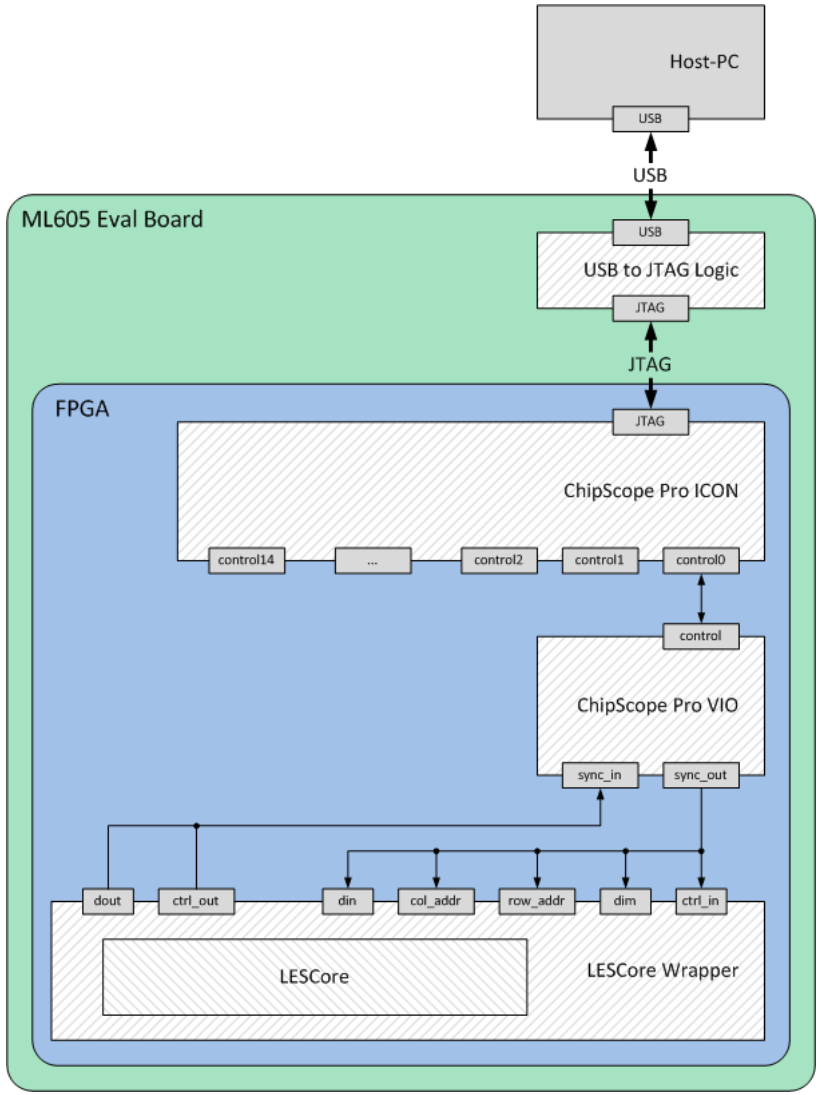
- Floating-Point Format
- Parallelität N
- Speichergröße



Test

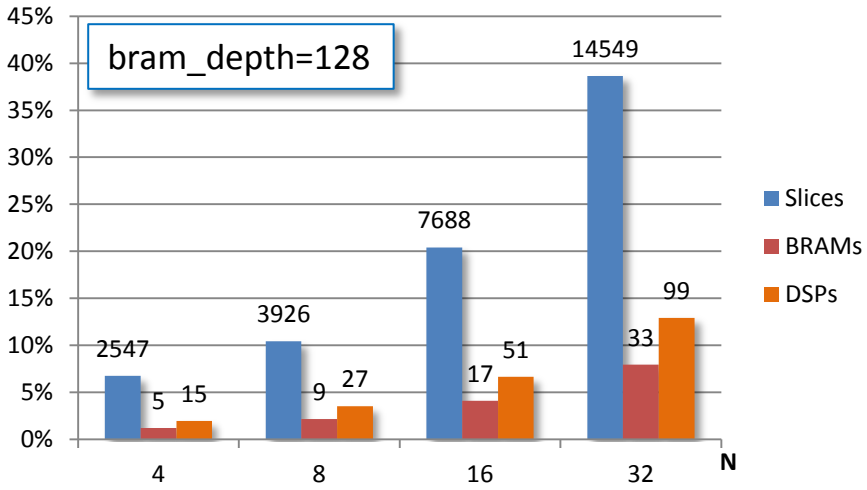
- Simulation mit ModelSim

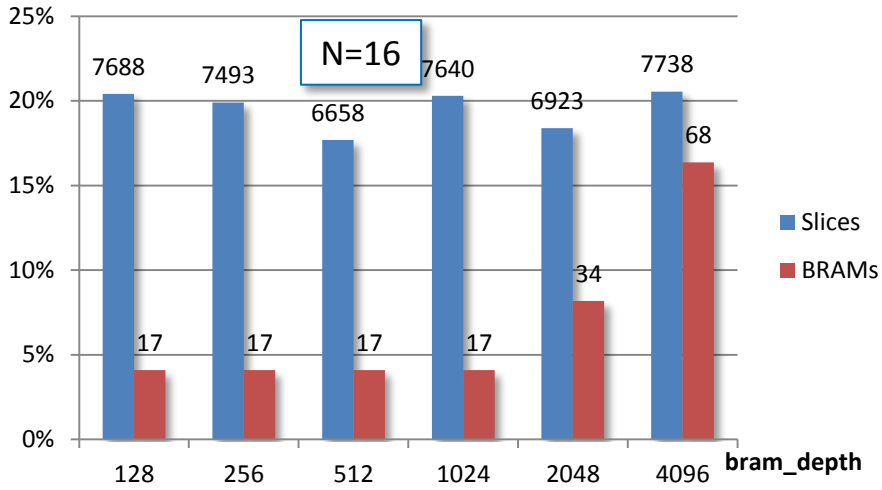
- Prototyp auf Xilinx® Virtex 6 FPGA
  - Zusätzliche Module für Kommunikation zwischen Host-PC ←→ FPGA
    - ChipScope Pro
    - Integrated Controller (ICON)
    - Virtual I/O (VIO)

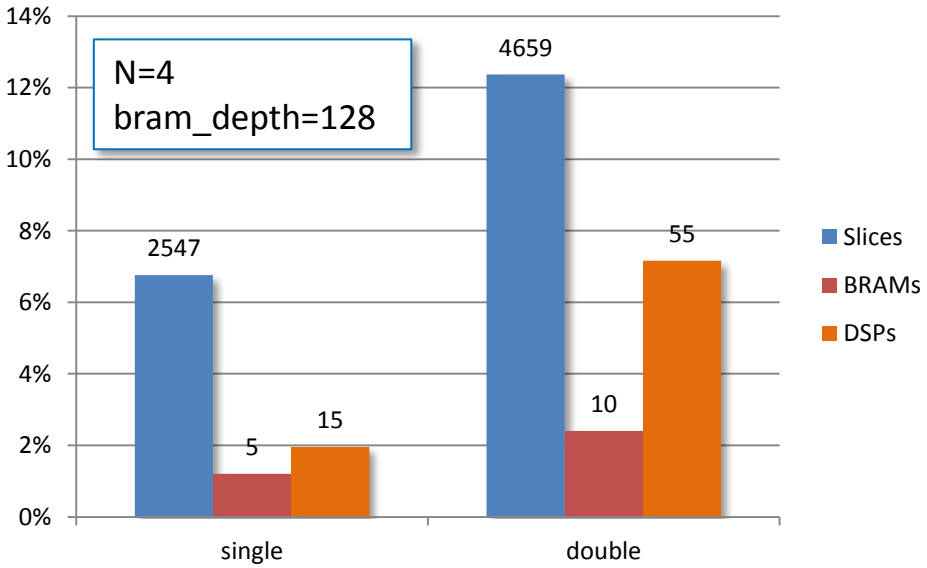


Auswertung

- FPGA Ressourcen (Virtex 6 VLX240T-1)







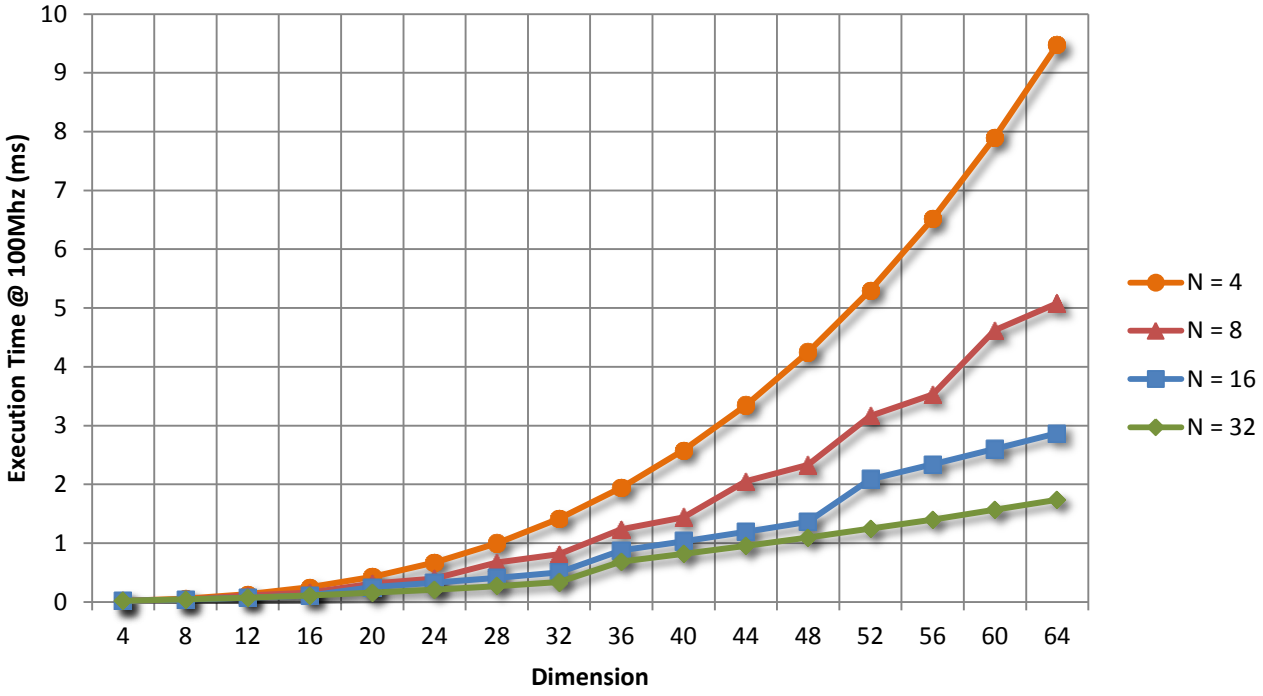
Auswertung

- Erreichbare Taktrate:
  - Virtex 6 VLX240T-1
  - Inklusive Host-PC Interface (ChipScope Pro ICON + VIO)

Design	XST (Post Synthese)	PAR (Post Place & Route)
N=4, bram_depth=128, IEEE-754 single	164 MHz	187 MHz
N=32, bram_depth=2048, IEEE-754 single	156 MHz	82 MHz

Auswertung

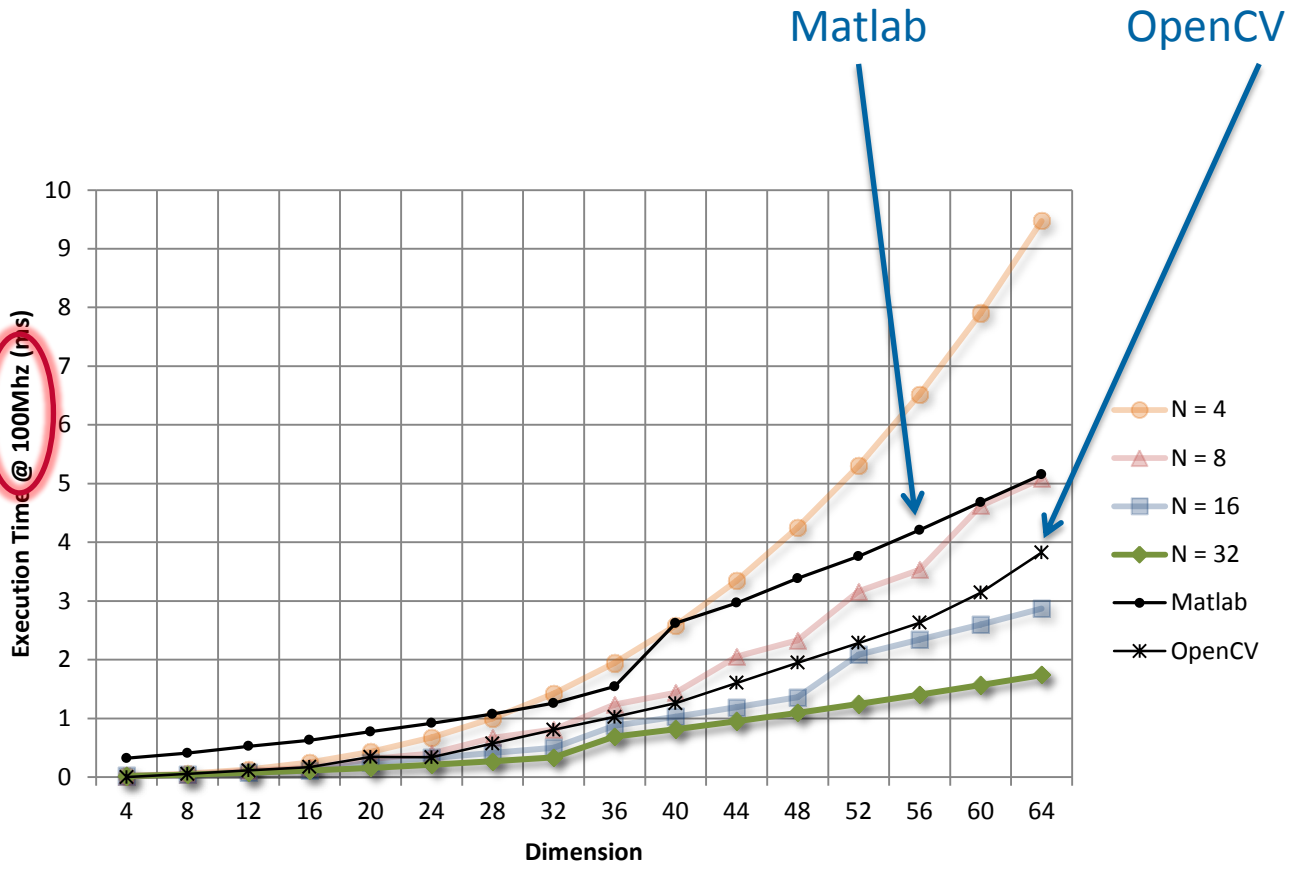
- Schlechtester Fall:
  - Immer Zeilenvertauschung
  - Alle Koeffizienten ≠ 0



Auswertung

- Schlechtester Fall:
  - Immer Zeilenvertauschung
  - Alle Koeffizienten ≠ 0

Vergleich mit Intel Core2Duo (1 Kern):



Zusammenfassung + Ausblick

- Flexibel, einfache Anpassung durch Parameter
- Einfache Schnittstelle (identisch mit BRAM Speicher)
- Speed-Up gegenüber Lösung mittels embedded Software bei Auslastung von nur 20% der FPGA-Ressourcen
- Fixed-Point Arithmetik
  - Deutliche Ressourceneinsparung
- Steigerung der Performanz
  - Großes Optimierungspotential in Zustandsautomaten (Handshakes)
  - Ausnutzung der Pipeline in Floating-Point IP-Cores