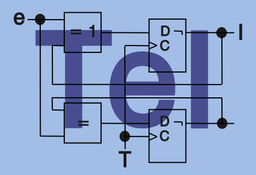
Untersuchungen zum Pipelining von DSP Operationen in einer rekonfigurierbaren Prozessordatenpfaderweiterung

Jan Schirok

s7642618@mail.inf.tu-dresden.de



Institut für Technische Informatik
http://www.inf.tu-dresden.de/Tel/

16.01.2007

Gliederung

- Motivation
- Pipelining
- Pipelining im ARRIVE
- Ergebnisse
- Ausblick

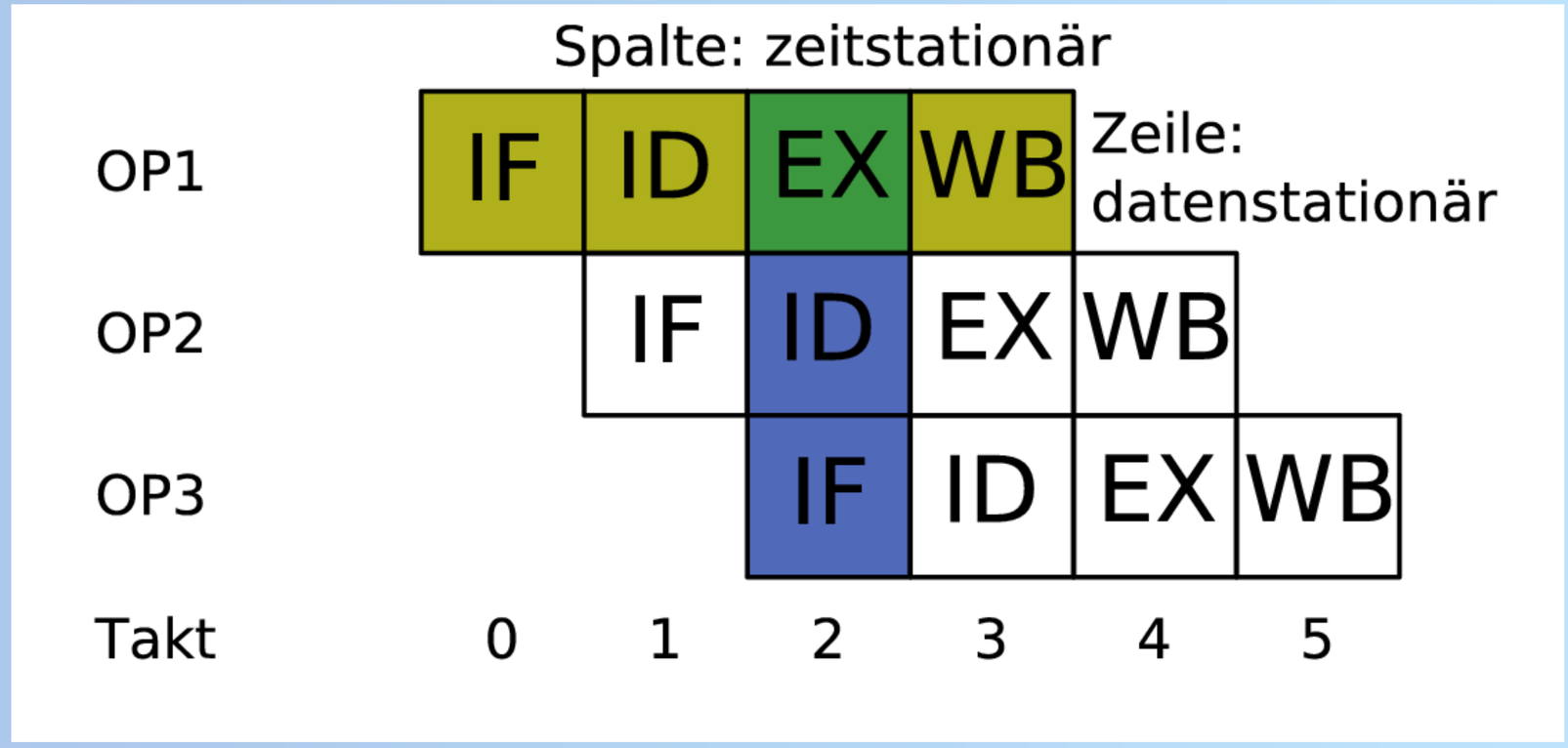
Motivation

- Anforderungen an digitale Signalverarbeitung
  - hoher Durchsatz
  - geringe Latenz
  - geringer Leistungsverbrauch
  - geringe Kosten
  - hohe Flexibilität
- Verwendung (grobgranular) rekonfigurierbarer Architekturen
- Erhöhung von Taktrate und Durchsatz: Pipelining

Pipelining

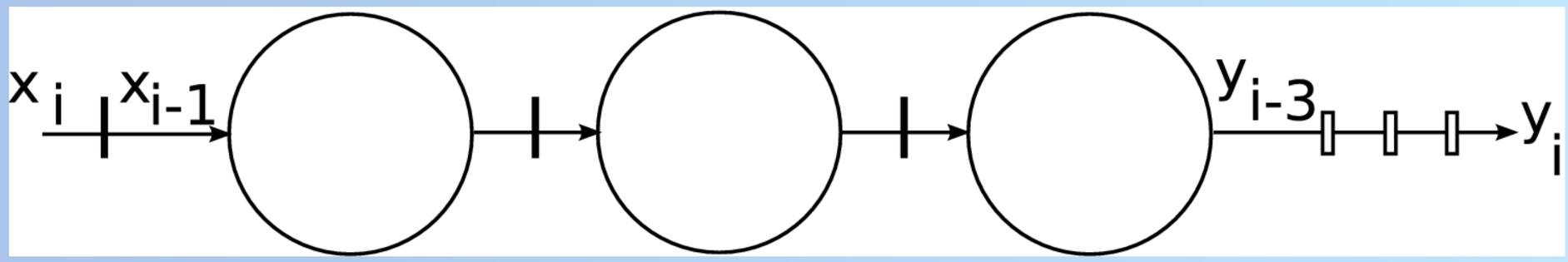
Pipelining in Prozessoren

- Aufteilung des Befehls, z.B.
  - IF – Instruction Fetch
  - ID – Instruction Decode
  - EX – Execute
  - WB – Write Back
- Phasen unterschiedlicher Befehle zeitlich parallel



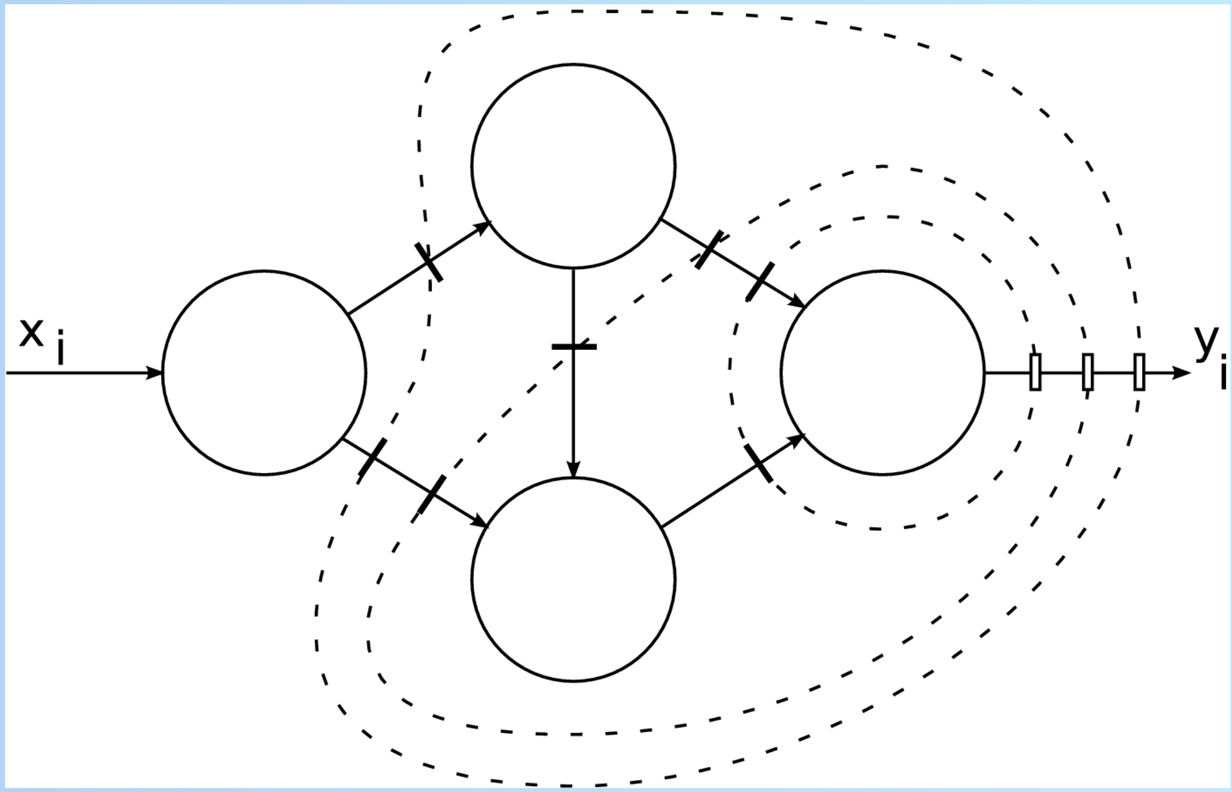
Umsetzung des Pipelining in Hardware

- für jede Pipelinestufe: eigene Abarbeitungseinheit
- Register zwischen den Stufen
- in einem Takt Abarbeitung mehrerer aufeinanderfolgender Befehle



Pipelines in DSP Algorithmen

- Einzeloperationen pipelinebar
- grafische Lösung
  - „fully pipelined“
  - kritischer Pfad abhängig von längster Operation

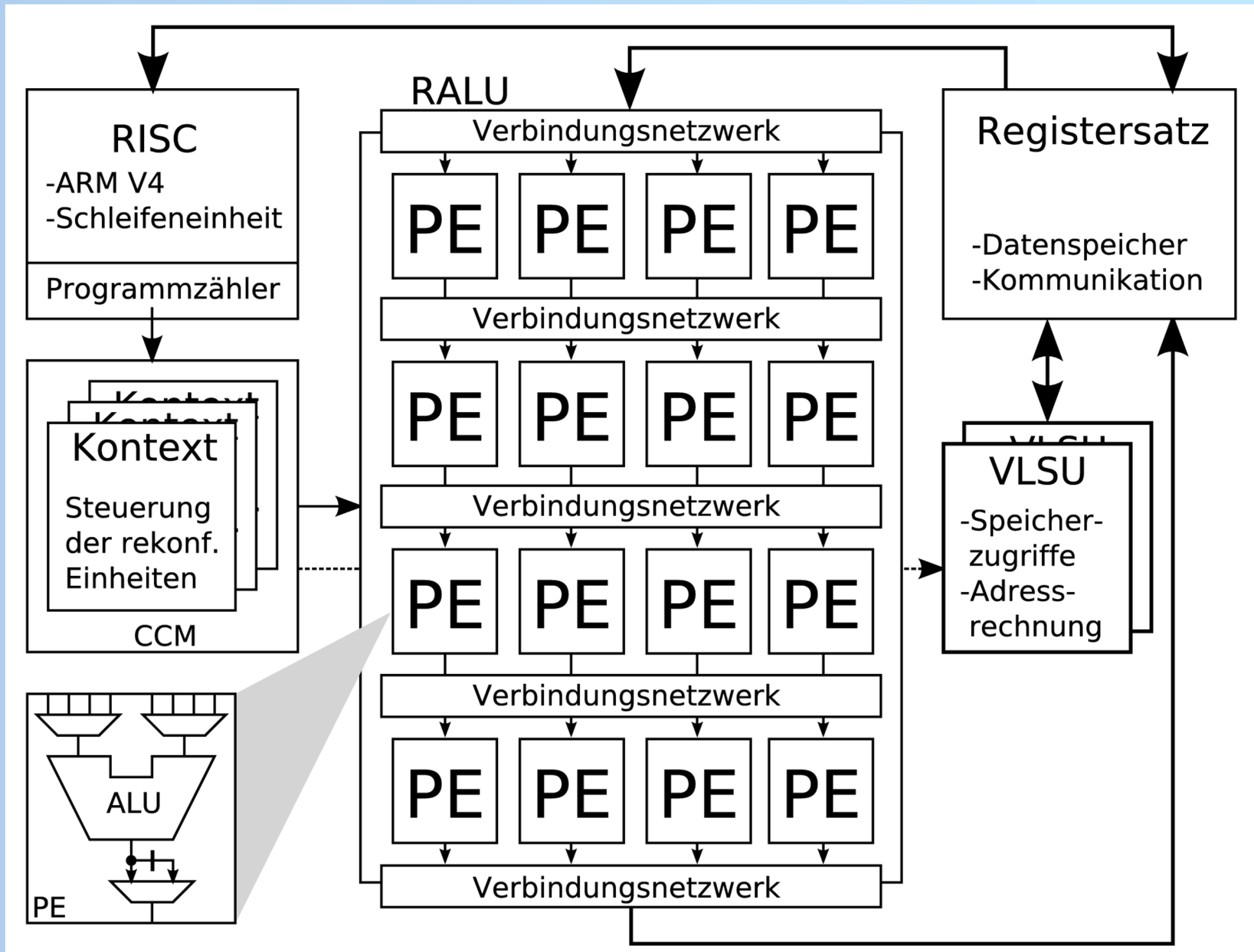


- Lösung ungeeignet für Schleifen
  - Schleifentransformation
  - Softwarepipelining

ARRIVE

ARRIVE - Allgemein

- grobgranular rekonfigurierbare Architektur
- Prozessordatenpfad-erweiterung
- TADL Modell
- VHDL Modell
- FPGA Prototyp
  - Debug-Interface
- ASIC Design
  - Standardzellen
  - 0,18um UMC
  - Leistungsverbrauch simulativ ermittelbar
- Assembler (zeit-stationär)



Pipelines in ARRIVE

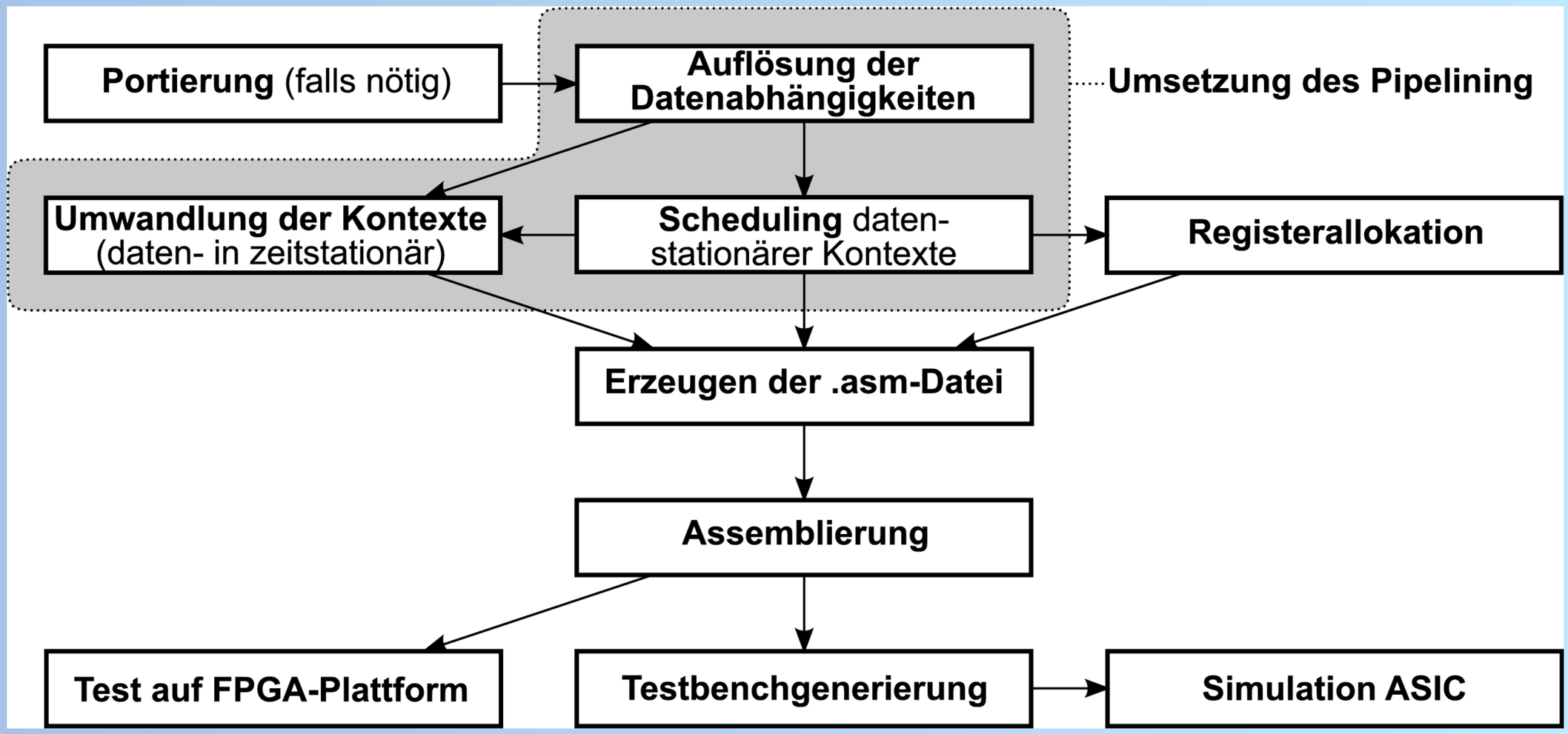
- RISC Pipeline
  - 3-stufig: IF, ID, EX
  - transparent, immer aktiv
- VLSU Pipeline
  - Adresse vorausberechnet
  - Write-Queue
  - transparent, immer aktiv, Hazards möglich
- RALU Pipelining
  - aktivierbares Ausgangsregister für jede PE
  - konfigurationsabhängig
  - Ansatzpunkt für Algorithmen

Pipelining der RALU

- Normalfall in der RALU
  - Abarbeitung Lesen+Verarbeiten+Schreiben in einem Takt
  - langer kombinatorischer Pfad
  - nur geringe mögliche Taktraten
- durch Konfiguration aktivierbare Ausgangsregister an PEs
  - Unterbrechung des kritischen Pfades
  - Pipelinestufe = in kompletter Zeile Ausgangsregister aktiviert
- Möglichkeit: Register dauerhaft aktivieren
  - Maximallänge der kritischen Pfade verringert
  - verringerte Flexibilität
  - bei anderen Konfigurationen (8x8 statt 4x4) von größerer Bedeutung

Ablauf Pipelining

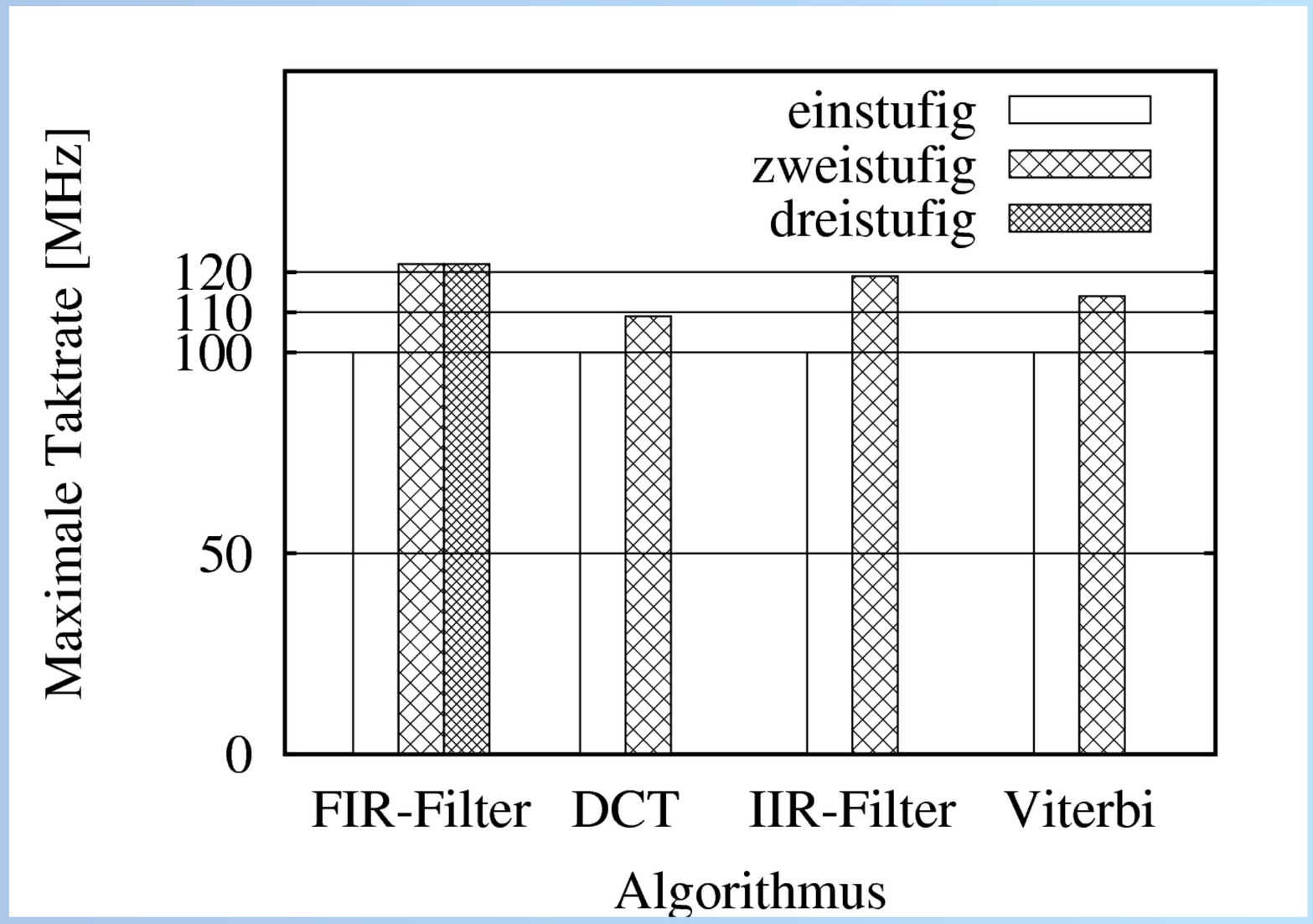
- allgemeines Vorgehen
- umgesetzt für 4 Algorithmen (FIR, IIR, DCT, Viterbi)
- Ansätze zur Automatisierung



Ergebnisse

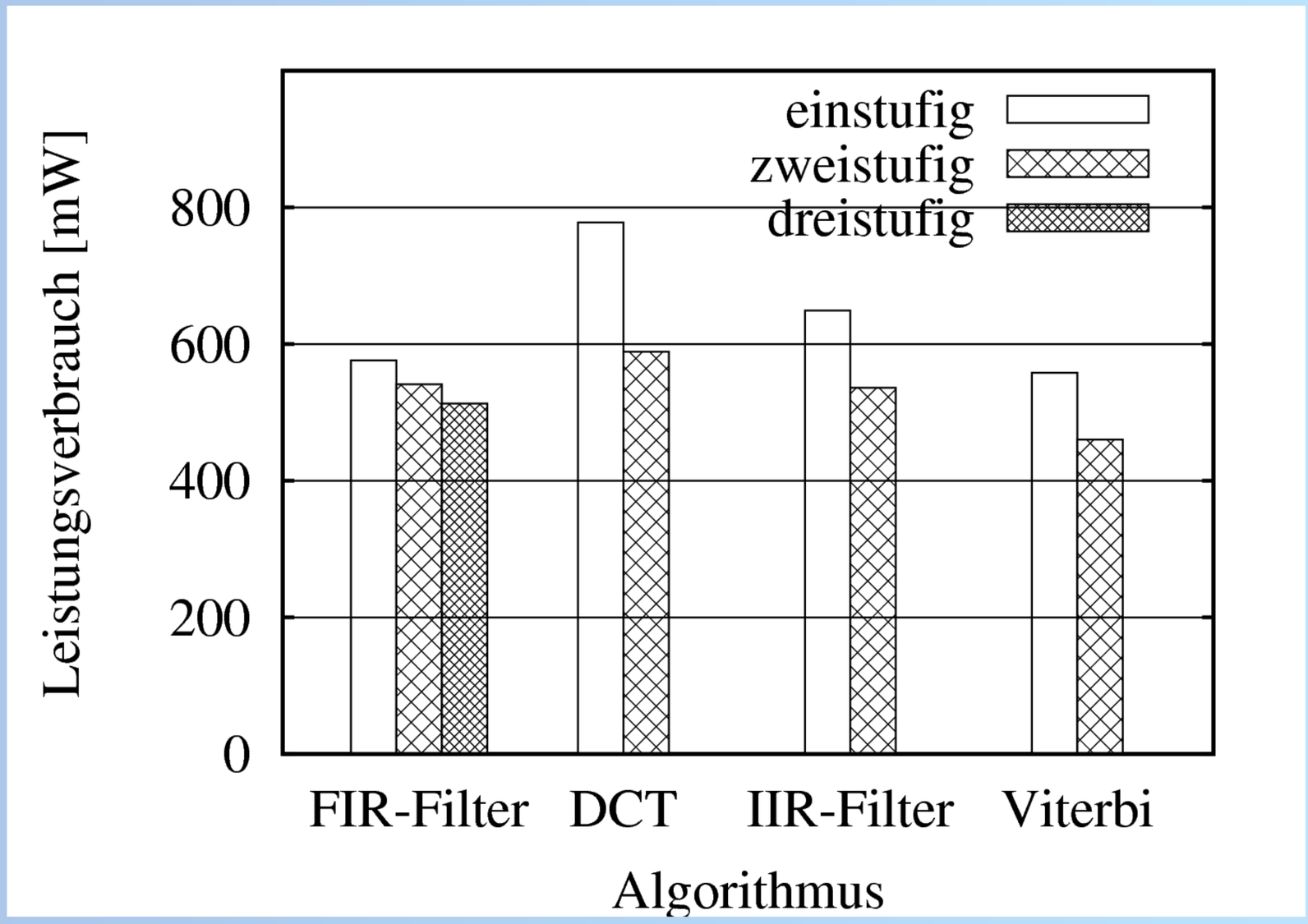
Ergebnisse – Erzielte maximale Taktrate

- erzielte Taktrate steigt mit Einsatz vom Pipelining
- Taktrate bei zwei-/dreistufiger Variante vergleichbar
- von Faktor 2 weit entfernt



Ergebnisse – Ermittelter Leistungsverbrauch

- Leistungsverbrauch sinkt mit Einsatz des Pipelinings
- Grund: verringerte Schaltaktivität mit aktivierten PE-Registern



Ergebnisse

- Erhöhung der erzielbaren Taktrate
  - mit Einsatz von Pipelining steigt erzielte Taktrate
  - „Sättigungseffekt“ mit mehr als zwei Pipelinestufen
  - Maximalfrequenz etwa 120 MHz
- Verringerung des Leistungsverbrauchs
  - mit Einsatz von Pipelining verringert sich der Leistungsverbrauch
  - weniger Schaltaktivitäten durch Registereinsatz
  - Erhöhung auf dreistufige Pipeline von Vorteil
- nur zweistufige Pipeline vorteilhaft
  - Taktrate begrenzt
  - Latenz
  - Auslastung
  - Abhängigkeiten

Ausblick

- Abhängigkeiten der Ergebnisse von der Konfiguration der RALU
  - Untersuchung anderer Konfigurationen
  - Redesign aller Algorithmen notwendig
- Zentraler lokaler Registersatz = Flaschenhals im ASIC Design
  - Dezentralisierung von Adressregistern der VLSUs
  - weniger enge Kopplung von RISC und Registern
  - sinnvoll ab Taktfrequenzen > 100 MHz
- (Teil-)Automatisierung des Pipelining

Ende

Danke für die Aufmerksamkeit