TECHNISCHE UNIVERSITÄT DRESDEN

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

Realisierung eines Speichermanagements zur Zugriffsvirtualisierung von konkurrierenden Nutzerdesigns auf rekonfigurierbarer Hardware

Verteidigung der Studienarbeit

Alexander Kemnitz
Alexander.Kemnitz1@mailbox.tu-dresden.de

Dresden, 22.9.2016

Inhalt

1 Einleitung

Motivation

- FPGA auf Cloud
  - Cloud-Computing gewinnt an Bedeutung
  - Zurzeit vor allem CPUs und GPUs
  - FPGAs bieten flexible Hardware
- Mehrere Programme pro FPGA

  → hohe Auslastung der Ressourcen

=> Framework für Einbettung von FPGAs in Cloud

1 Einleitung

RC²F

- Host - Device Prinzip
- Verwaltung durch RC2F Core
- 1 - n virtuelle FPGAs als Recheneinheiten
- 1 Speichercontroller als Schnittstelle zum RAM

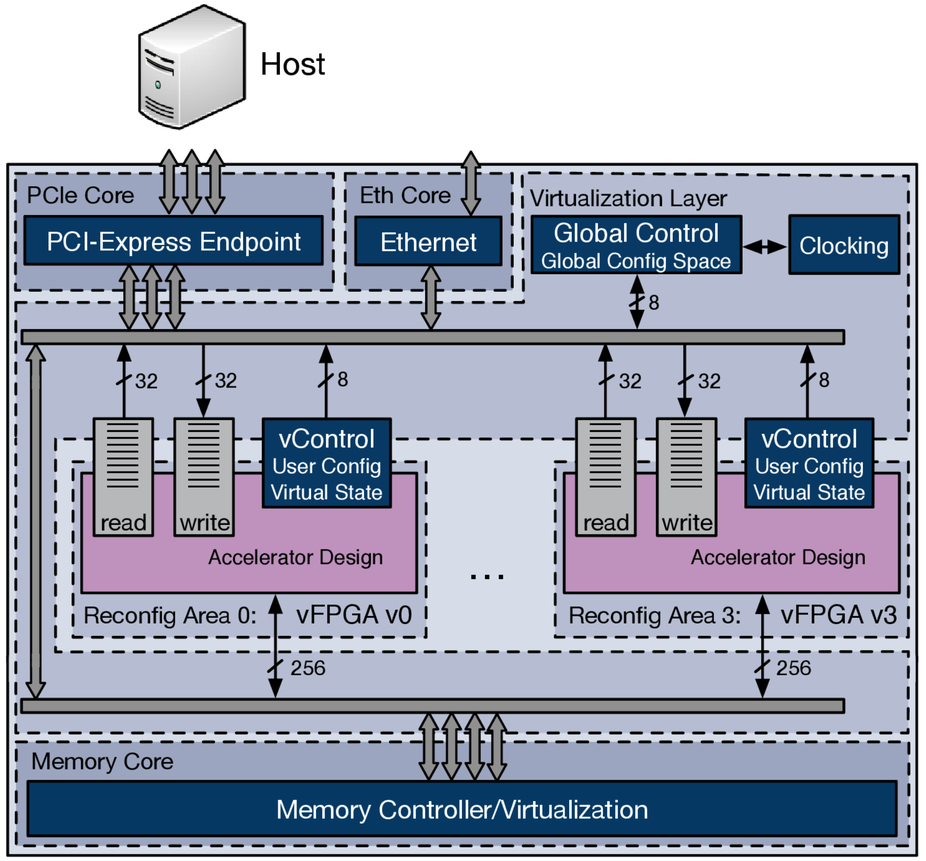

Abb. Konzept des Reconfigurable Cloud Computing Framework [1]

1 Einleitung

Aufgabenstellung

- Schnittstelle:
  - Bis zu 4 Nutzerdesigns
  - Zugriff von Host
- Ansteuerung des RAMs
- Getrennte Speicherbereiche

  $\rightarrow$ Virtualisierung
- Konkurrierende Zugriffe

  $\rightarrow$ Arbitrierung
- Hohe Bandbreite
- Nutzerfreundliches Interface

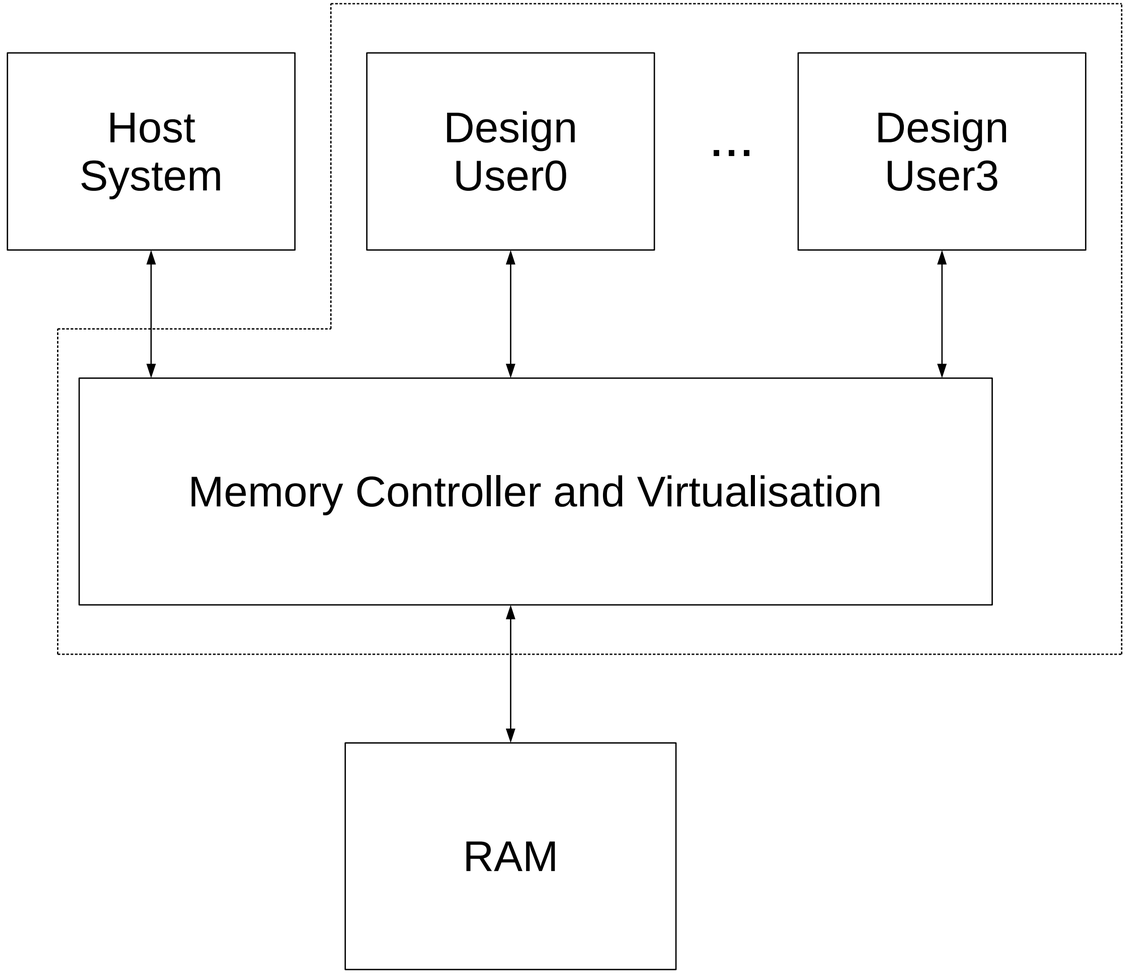

Abb. Vereinfachte Darstellung der Aufgabenstellung

2 Grundlagen

Speicherverwaltung

- Datensicherheit
- Virtualisierung (lineare Adressierung)
- Hohe Dynamik
- Mögliche Ansätze:
  - Statischer Speicherbereiche → keine Dynamik zur Laufzeit
  - Basis- und Grenzregister → wenig Dynamik zur Laufzeit
  - Zugriffsrechte Blöcke → keine Virtualisierung
  - Seitentabellen → geeignete Lösung

2 Grundlagen

Seitentabelle

- Aufteilung des Speichers
  in Blöcke (Seiten)
- höchstens 1 Nutzer pro Block
- Je Nutzer eine Tabelle
- 1 Tabelle Zur Verwaltung
  → Host

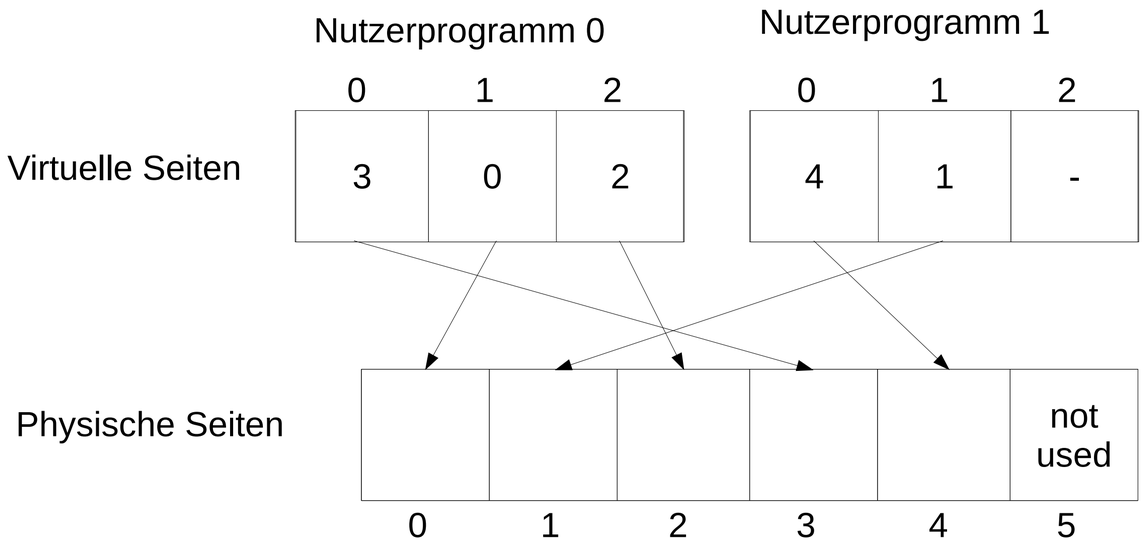

Abb. Grundprinzip der Seitentabelle

2 Grundlagen

Zugriffsverteilung

- Erhalt der Datenrate
  - → Blockweise Zugriffe
- Faire Aufteilung der Zugriffszeiten
  - → Round Robin
  - → Feste Blockgröße

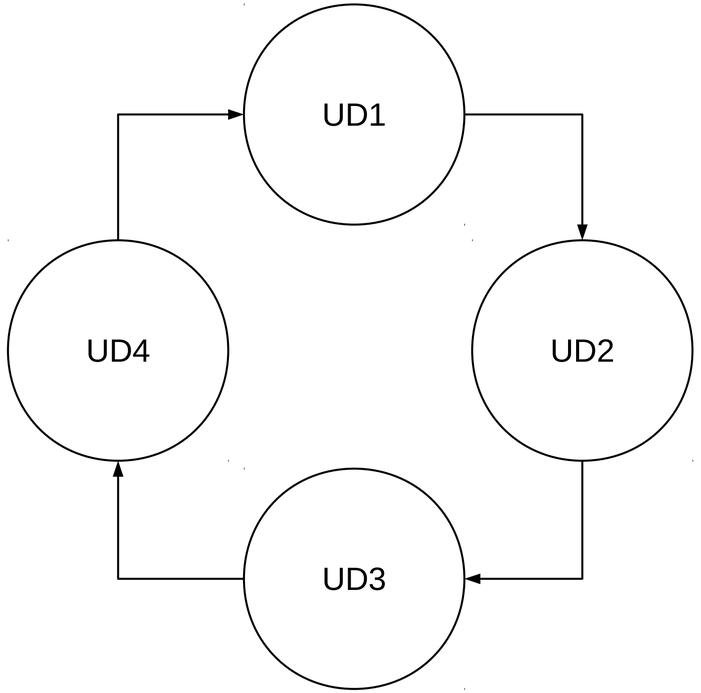

Abb. Prinzip des Round Robin für 4 UDs

2 Grundlagen

Physische Schnittstelle

- RAM muss physisch angesteuert werden
- IP Core: Memory Interface Generator
- 2 Interfaces:
  - AXI : ARM AMBA Industrie Standard
  - User Interface (einfach)
- UI : einfach und nicht Xilinx gebunden

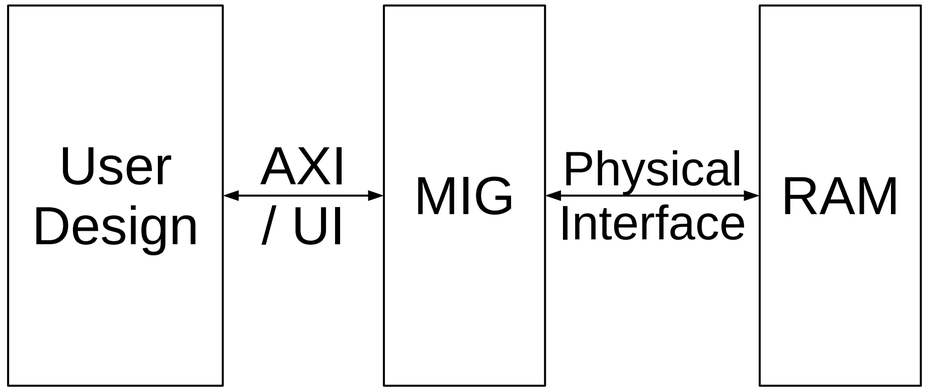

Abb. Vereinfachte Darfstellung des Xilinx Memory Interface nach [2]

3 Entwurf und Implementierung

Gesamtentwurf

- Pro Nutzerdesign 1 Seitentabelle
- 1 Queue
- 1 Switch
- Blockweise Zugriffe
- Aufteilung der Blockzugriffe in Teil-Bursts durch Wrapper

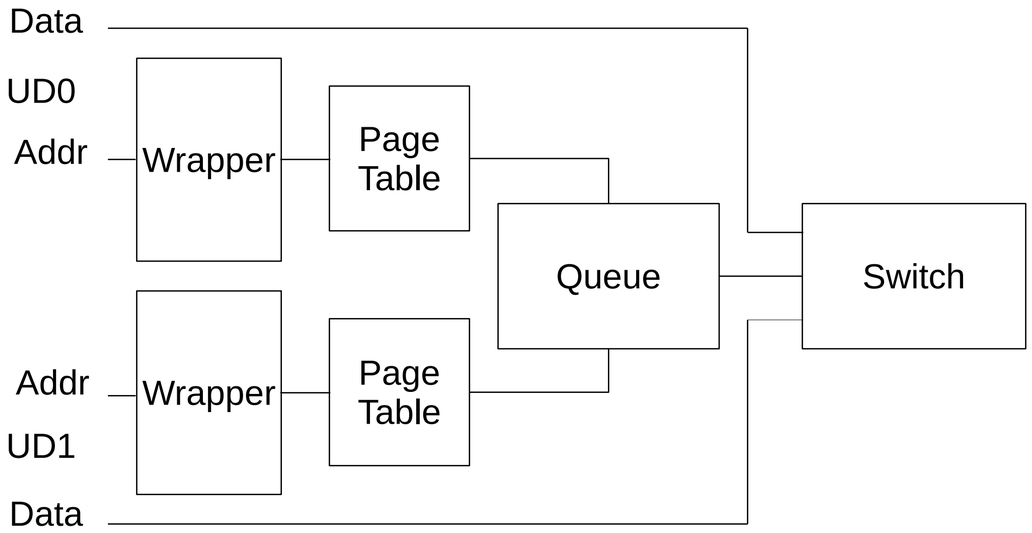

Abb. Funktioneller Entwurf des Gesamtdesigns

3 Entwurf und Implementierung

Ablauf eines Speicherzugriffs

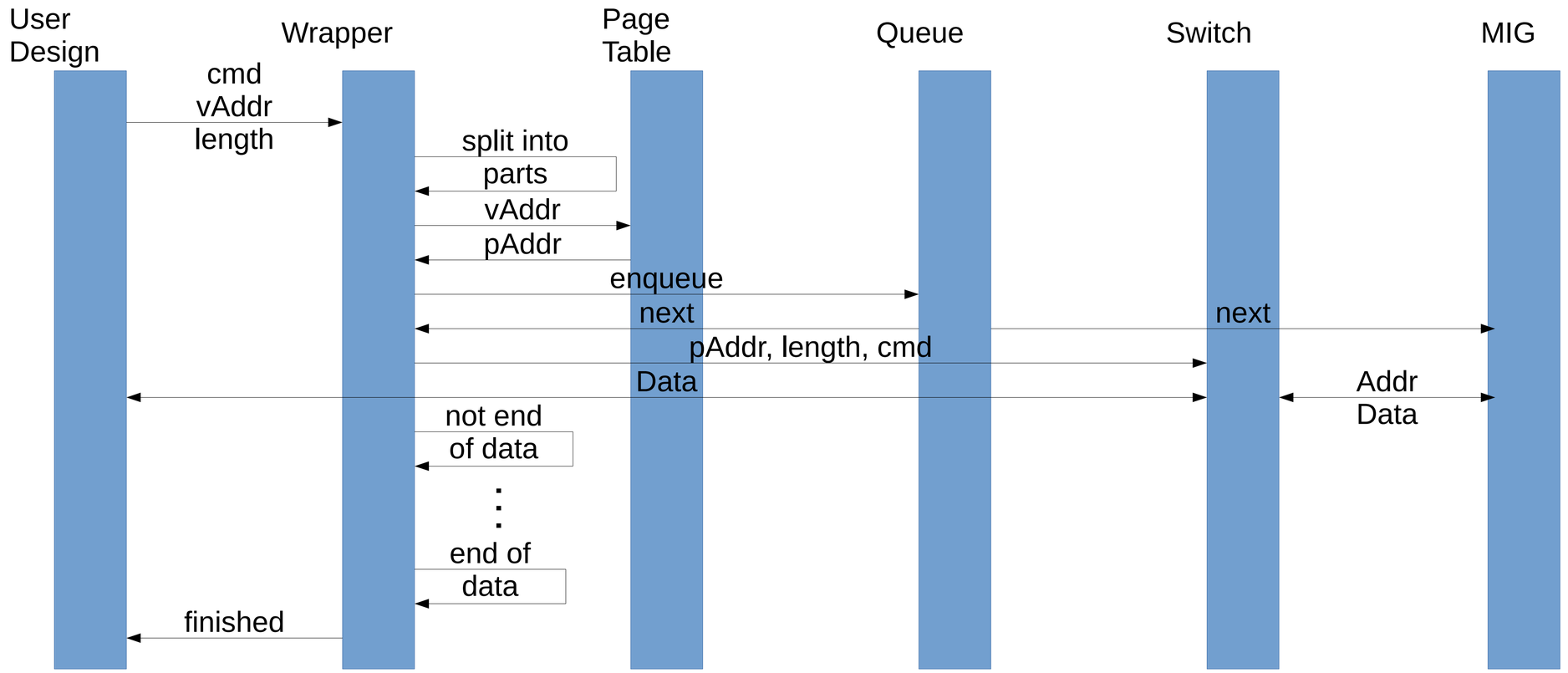

Abb. Ablauf eines Speicherzugriffs

3 Entwurf und Implementierung

Schnittstelle zum Nutzerdesign

- FIFO Interface für Daten
- Strobe für Befehl
  - Zugriffsart
  - Startadresse
  - Größe des Arrays

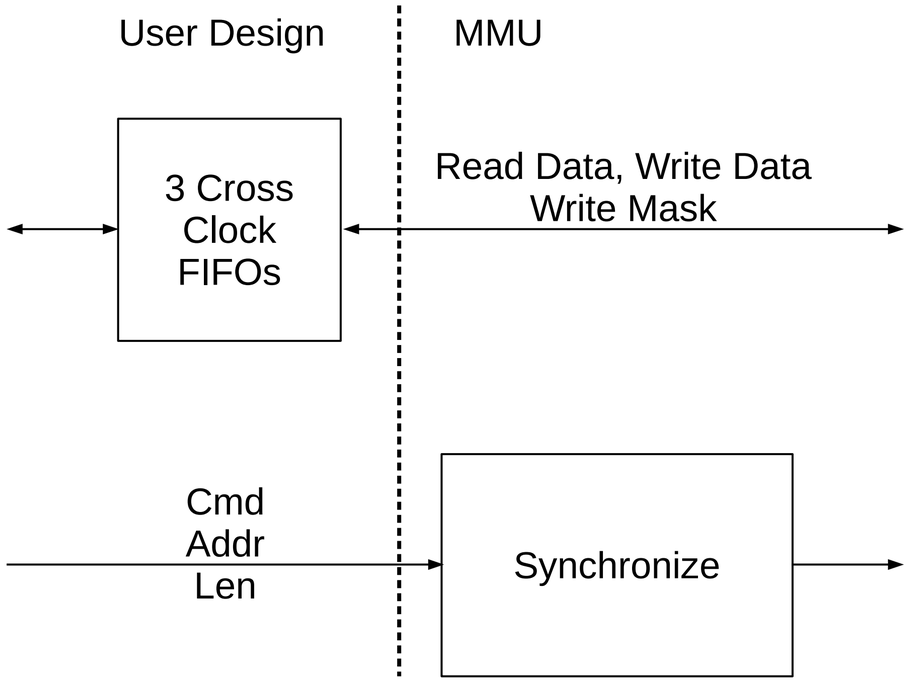

Abb. Grobe Darstellung der Schnittstelle zum Nutzerdesign

3 Entwurf und Implementierung

Queue

- FIFO für Nutzer

  → Round Robin ähnlich

  - Wrapper verhindert mehrfaches einreihen

- Priorisierung bei zeitgleichen Requests
- Register speichert nicht verarbeitete Reqests

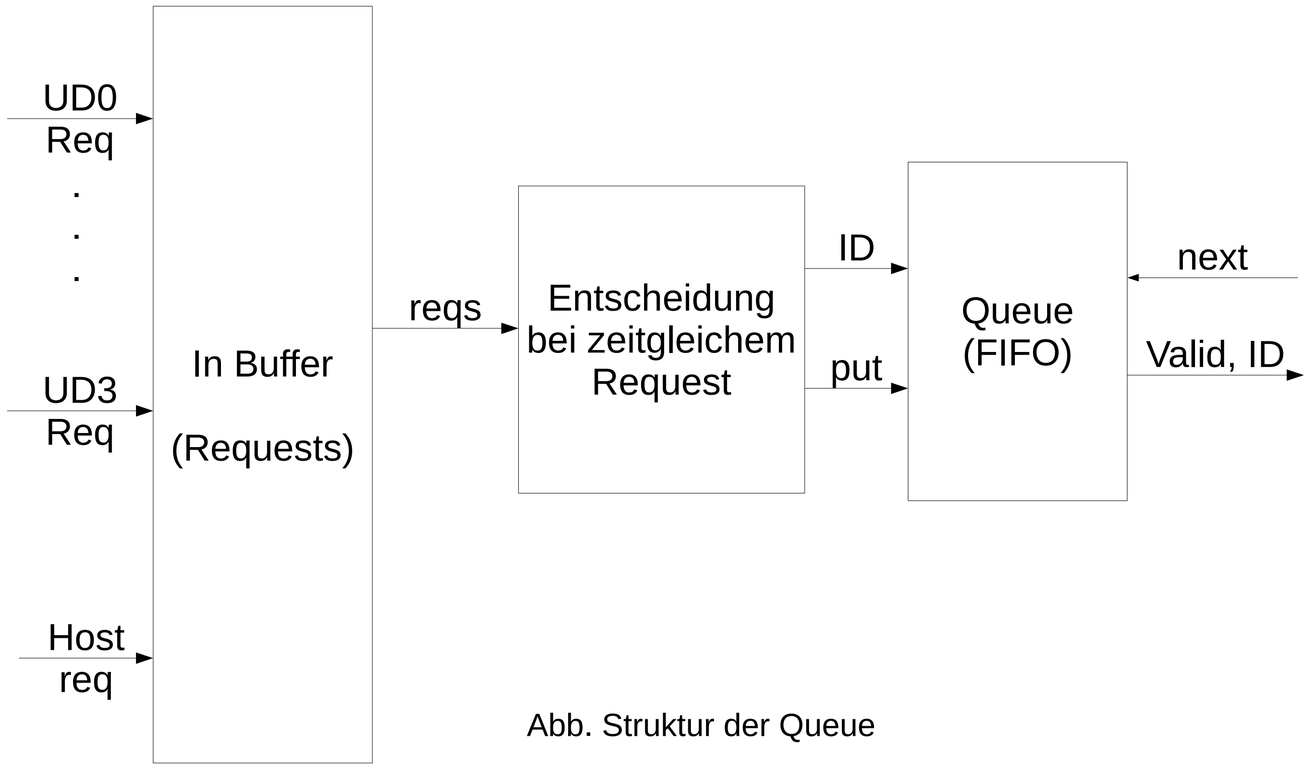

Abb. Struktur der Queue

3 Entwurf und Implementierung

Switch

- FSM für Steuerung
- Counter für Daten und Adresse
- MUX für Datenleitung
- Konvertierung zwischen

  FIFO Interface und UI

  → Zwischenspeichern von Lesedaten

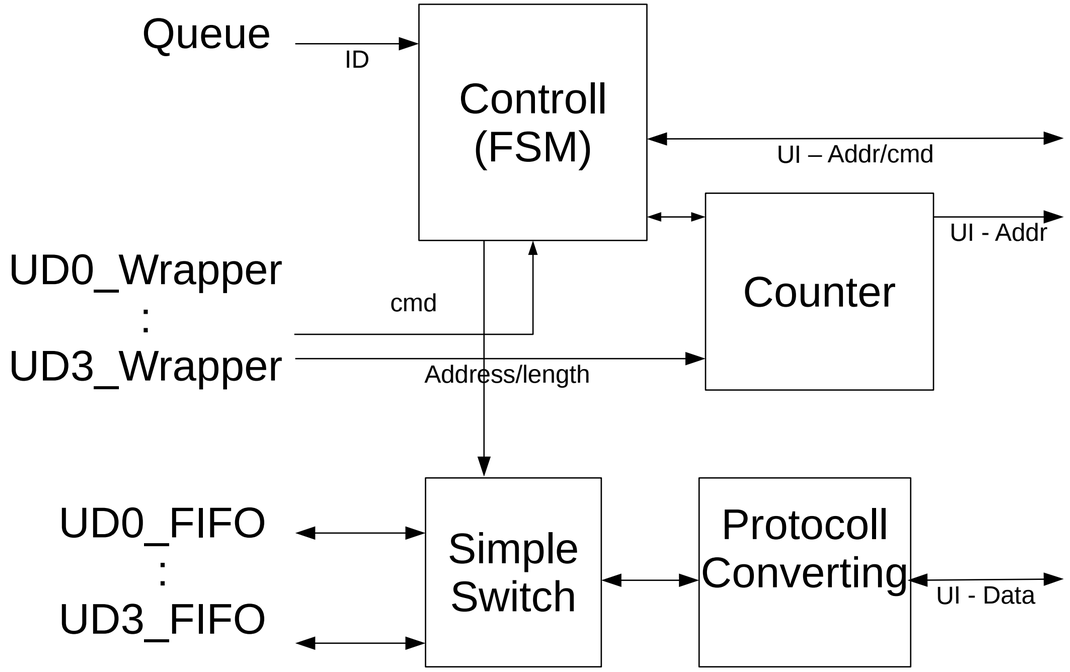

Abb. Grobe Struktur des Switch

3 Entwurf und Implementierung

Switch - FSM

- Calib: lock-Zustand
- Idle: Warten auf ID
- Start access: Laden von Operation und Adresse
- Read/write
- Collect data: Warten bis das Nutzerdesign
  alle Lesedaten erhalten hat
- Completed: Strobe an Wrapper

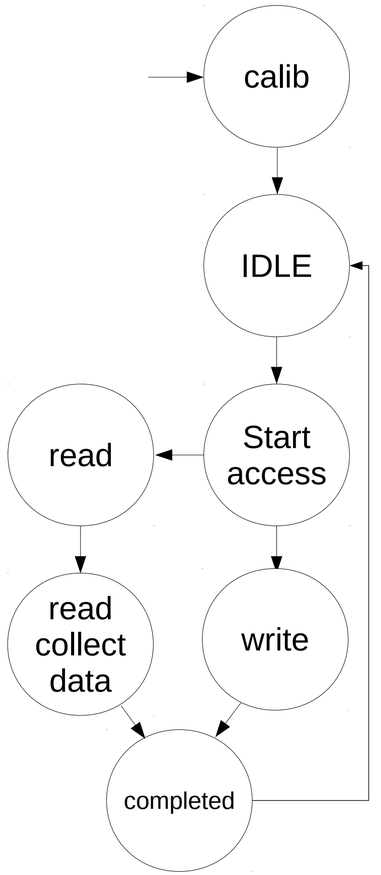

Abb. Skizze der FSM des Switch

3 Entwurf und Implementierung

Freiheitsgrade / Parameter

- Größe des Lesezwischenspeichers
  - Ressourcen vs Datenrate
  - Alternativ ohne Zwischenspeicher
    - → Mehr Verantwortung für UDs
- Größe der Burstzugriffe
  - Benachteiligung kleiner Zugriffe vs Datenrate
- Seitengröße
  - Ressourcen vs Dynamik

4 Ergebnisse

Zwischenspeicher für Lesezugriffe

- Verhindern von Wartezyklen
- Größe von 32 reicht aus
- Ressourcenbedarf zu vernachlässigen

  → Vereinfachung der Schnittstelle

  zu geringem Preis

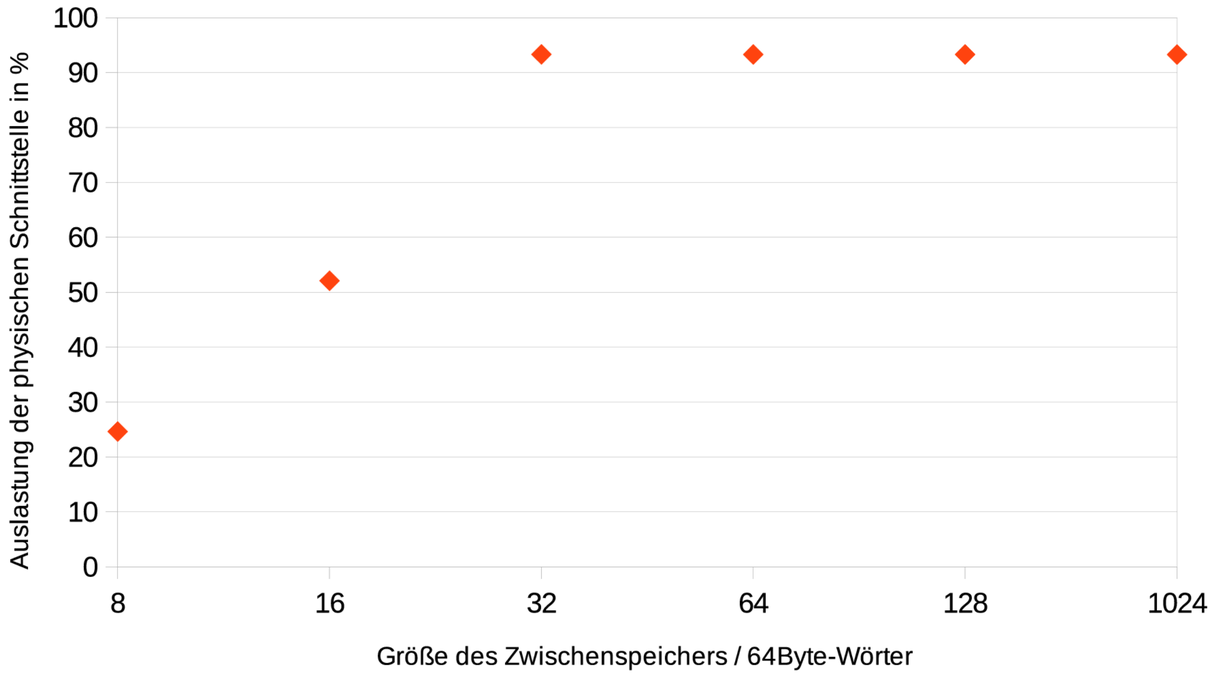

Abb. Messung der Datenrate für Lesezugriffe
Variation der Größe des Zwischenspeicher für Lesedaten des Moduls Switch

4 Ergebnisse

Größe der Burstzugriffe

- Typische Größenordnung

  ca. 10-100 KiB / 8-64 KiB

  → Absolutes Maximum
- Verkraftbare Datenrate ab 32 KiB
- Aktuelles Problem: Sammeln

  der Lesedaten blockiert
- Schreiben wäre ab 4-16 KiB gut

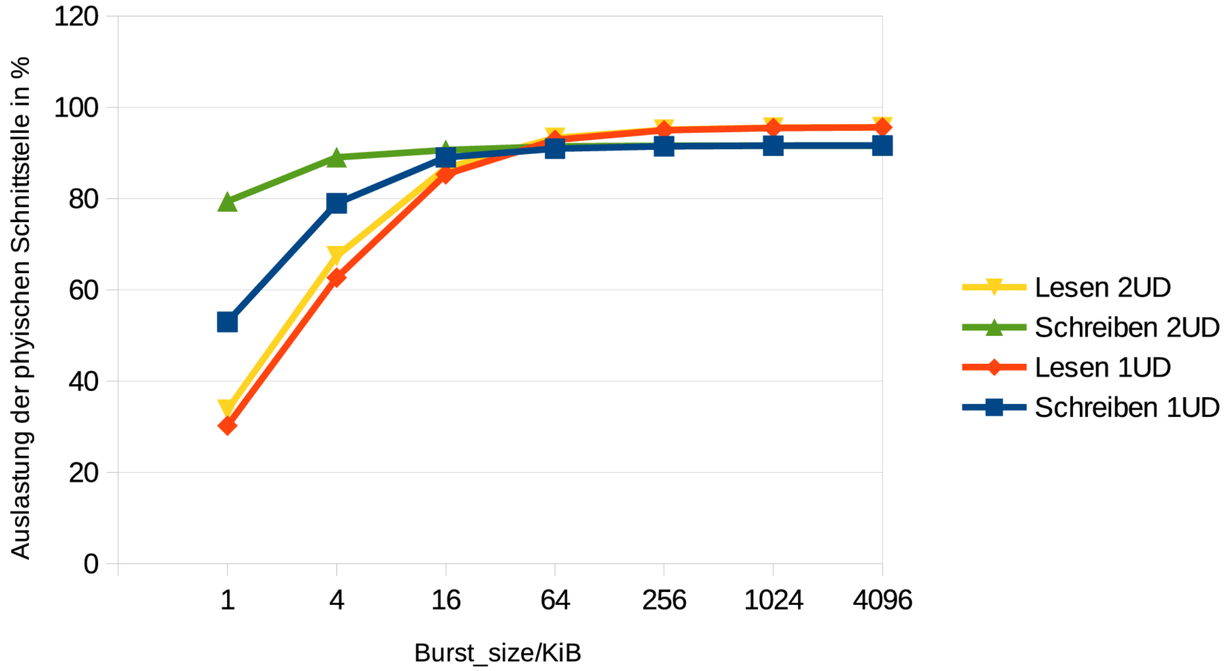

Abb. Messung der Datenrate für Lesezugriffe und Schreibzugrffe
Variation der Burstgröße

4 Ergebnisse

Größe der Burstzugriffe – getrennte Taktdomänen

- MMU 200 MHz
- 1 UD 150 MHz
- 1 UD 100 MHz
- Vergleichbare Kurven
- Ab 8-16 KiB gute Werte

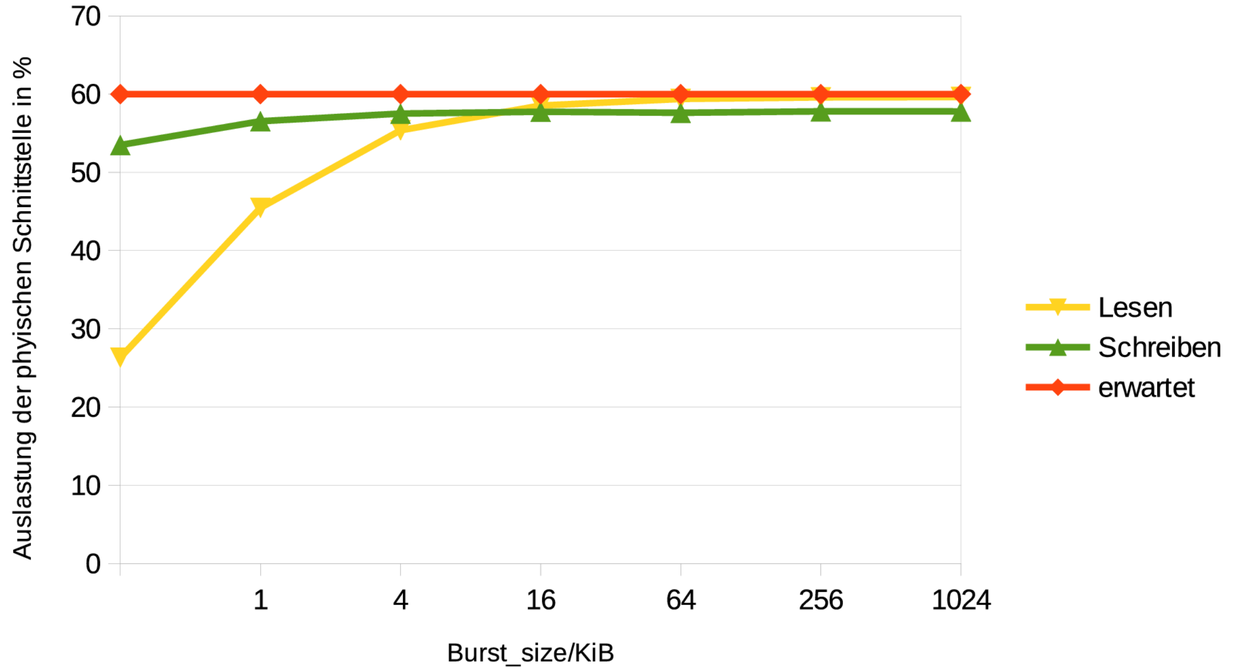

Abb. Messung der Datenrate für Lesezugriffe und Schreibzugrffe
Variation der Burstgröße
1 UD 100 MHz, 1 UD 150 MHz, normale Crossclock-FIFOs

4 Ergebnisse

Page Table

- Verkraftbarer Ressourcenbefarf: ca 1-2%

  → 0,25%-0,5% pro Seitentabelle

- Messung 1 PT
- 2048-4096 Seiten

  / Seitengröße von 1-2 MiB

  → gute Dynamik

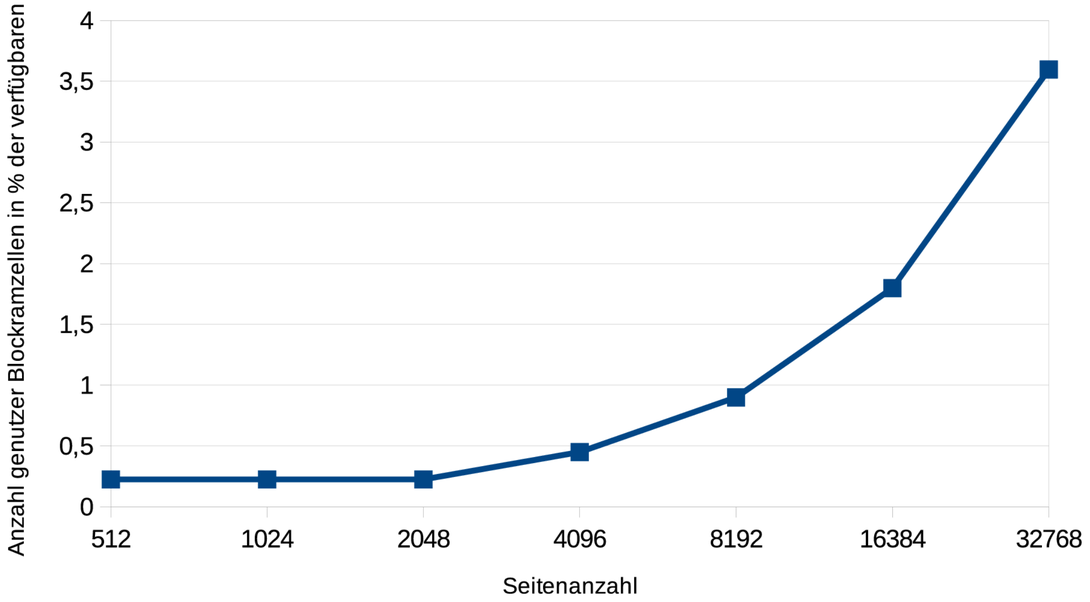

Abb. Für Seitentabelle benötiger Block RAM
Variation der Seitenanzahl

4 Ergebnisse

Latenz

- Minimale Dauer zwischen Initialisierung des Speicherzugriffs und Beginn Zugriffs ohne Synchronisierung:

  10 MMU-Taktperioden bzw. 50 ns (6 Wrapper, 3 Queue, 1 Switch)

- Maximale Dauer für 4 UDs bei voller Auslastung der MMU bei 64 KiB Burstgröße

  Ca. 4500 Takte bzw. 22,5 µs

4 Ergebnisse

Ressourcenbedarf

- Bis zu 2 UDs und der Host
- Zwischenspeicher:

  32 Datenwörter
- Burstgröße: 64 KiB
- Seitengröße 1 MiB
- MIG entscheidend

	MMU 1 UD	MMU 2 UD	MMU 2 UD + Host	MIG	Gesamt 2 UD + Host
Slices	0,14%	0,19%	0,74%	6,98%	7,72%
Slice Regs	0,032%	0,047%	0,061%	1,66%	1,72%
LUTs	0,078%	0,12%	0,39%	4,37%	4,75%
LUT RAM	0,006%	0,006%	0,006%	2,37%	2,38%
BLOCK RAM	0,45%	0,90%	0,90%	0%	0,90%

Tabelle: Ressourcenbedarf des Designs
als Anteil der vorhanden Ressourcen auf dem Board KC705
Variieren der angeschlossenen Designs

5 Schlussteil

Zusammenfassung

- Datendurchsatz: ca. 6 GB/s
- Ressourcenbedarf hängt vor allem vom MIG ab
- Grundlage für Datensicherheit
- Fairness der Zugriffsaufteilung ab Zugriffen von ca. 16-64 KiB
- Hohe Dynamik
- Einfaches Interface

5 Schlussteil

Ausblick

- Anpassen des Switch

  → Reduzierung der Burst-Größe

- Einarbeiten in das RC2F

  - Seitenverwaltung

  - Behandlung von Seitenfehlern und langsamen Designs

⇒ Erweiterung des RC2F für speicherintensive Anwendungsfälle

6 Anhang

Quellen

[1] Oliver Knodel und Rainer G Spallek (2015). “Computing Framework for Dynamic Integration of Reconfigurable Resources in a Cloud”. In:Euromicro Conference on Digital System Design (DSD). IEEE, S. 337–344

[2] XILINX 2015. Zynq-7000 AP SoC and 7 Series Devices Memory Interface Solutions v2.3

[3] XILINX 2015. 7 Series FPGAs Overview

[4] Tannenbaum, Andrew S. Carl Hanser und Prentice-Hall International 1995. Moderne Betriebssysteme

[5] Smith, James E und Nair, Ravi. Elsevier 2005. Virtual Machines - Versatile Platforms for Systems and Processes