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

Zwischenvortrag zur Studienarbeit

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

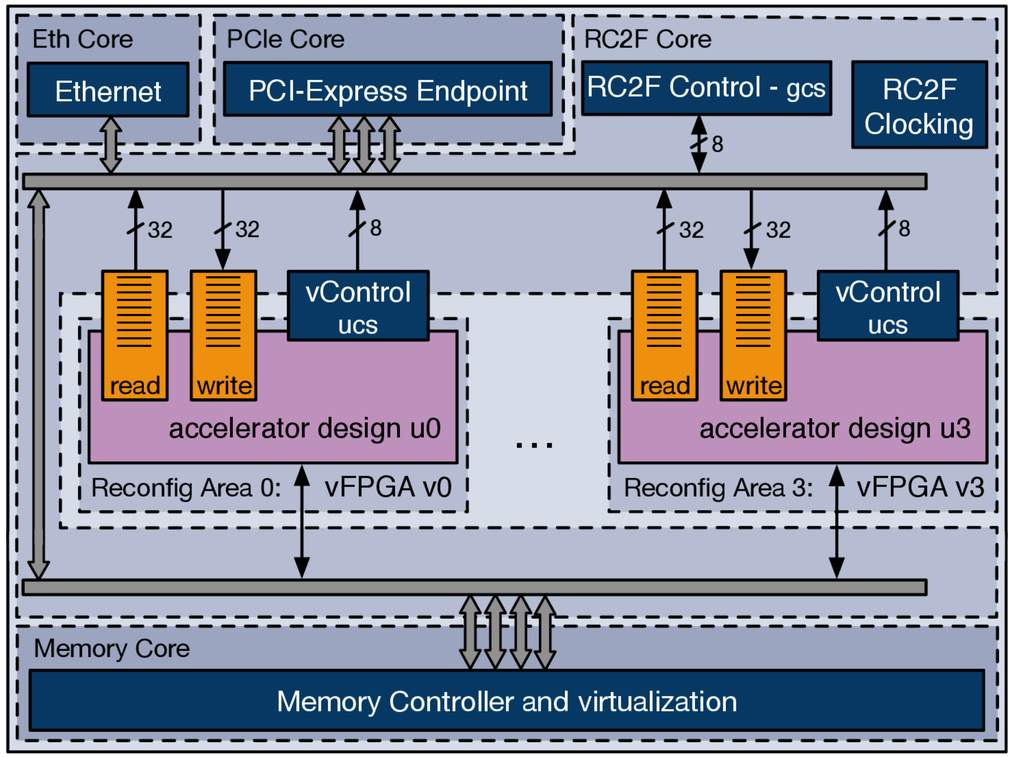
Dresden, 11.2.2016

Inhalt

1 Einleitung

Motivation

- FPGA auf Cloud
  - Cloudcomputing gewinnt an Bedeutung
  - FPGAs bieten Flexible spezielle Hardware
- Mehrere Nutzer pro FPGA
  - Hohe Auslastung der Ressourcen
- RAM als großer Zwischenspeicher



Konzept RC2F [1]

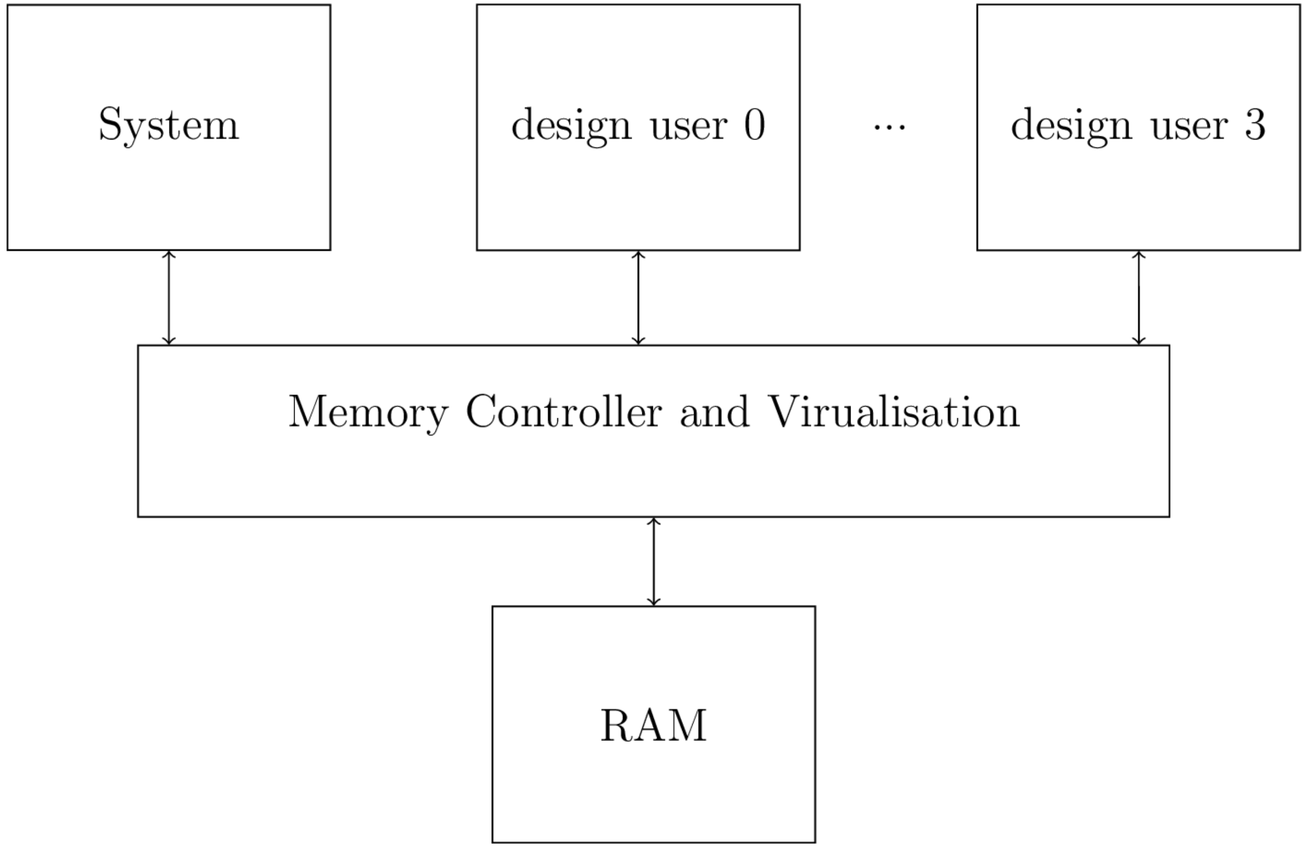
1 Einleitung

Aufgabenstellung

- Schnittstelle:
  - Bis zu 4 Nutzerdesigns
  - Zugriff über System
- Konkurrierende Zugriffe

  $\rightarrow$ Faire Behandlung
- Getrennte Speicherbereiche

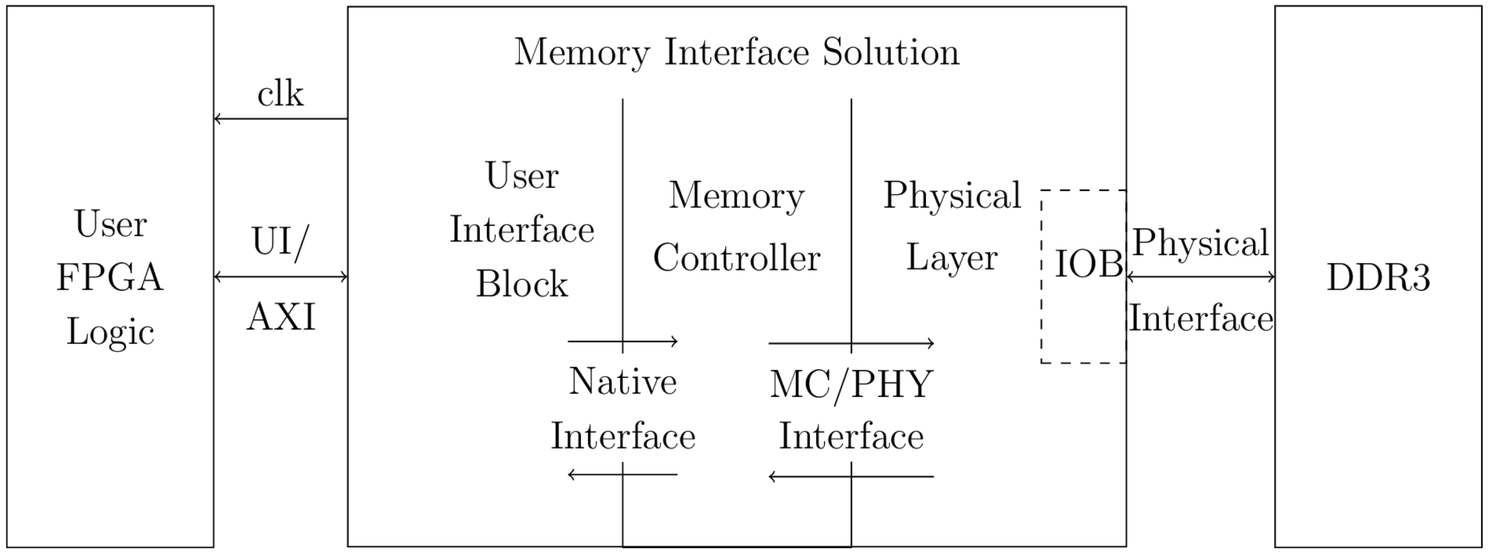
  $\rightarrow$ Virtualisierung
- Hohe Bandbreite



2 Speicher Controller

Memory Interface Generator

- Xilinx IP-Core
- 2 empfohlene Schnittstellen
  - UI (mit Beispiel)
  - AXI
- Eigene Taktdomäne

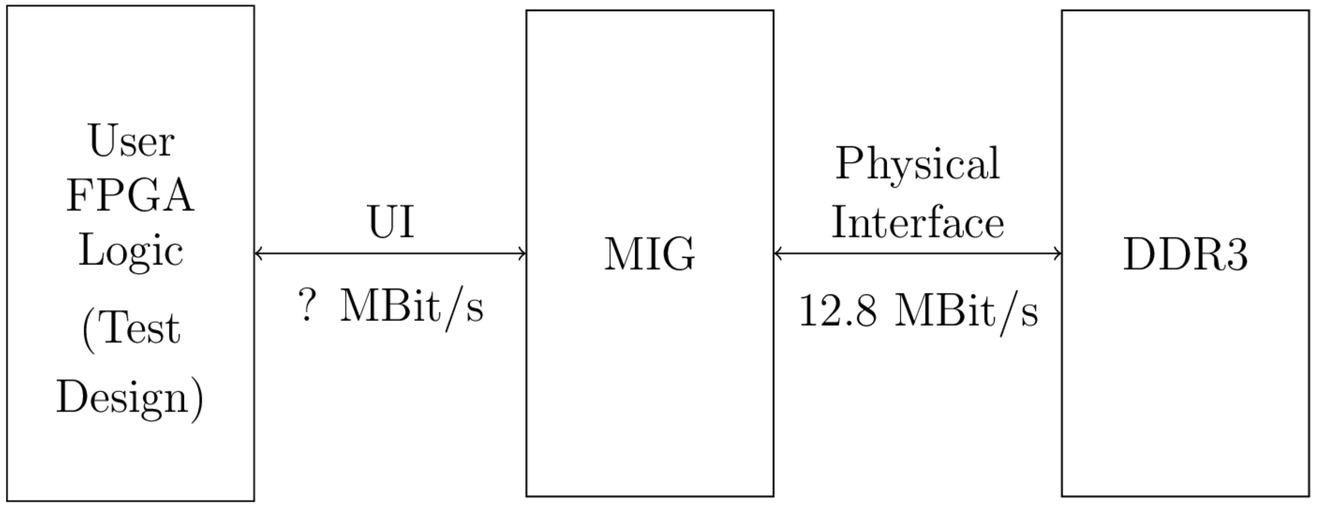


Vereinfachte Darstellung 7 Series FPGAs Memory Interface Solution (nach [2] Figure 1-51)

2 Speicher Controller

Test in der Simulation

- Datenrate
  - 1000 aufeinander folgende Adressen
  - Lesegeschwindigkeit : 88.5%
  - Schreibgeschwindigkeit: 86.2%
- Latenz
  - 1 Lese- und 1 Schreiboperation auf derselben Adresse
  - 30–48 Takte



Testaufbau zur Performance-Messung des MIGs

3 Virtualisierung

Begriff

- Trennen: Physisches Interface - Virtuelles Interface
  - Isomorphe Abbildung
  - Abstraktion
- Flexibel / Portabel

3 Virtualisierung

Speichervirtualisierung im Betriebssystem

- Zu jedem Programm ein Virtueller Adressraum
  - Größe unabhängig vom physischen Adressraum
  - Page Table: Virtuelle Adresse → Physische Adresse
- Eine Tabelle für das System
  - Zugriffsberechtigungen
  - In Benutzung
- Auslagerung der Seiten
- Fehlerbehandlung bei Seitenfehler

3 Virtualisierung

Eigenes Design

- Trennung des Speichers der einzelnen Nutzerdesigns
- Dynamische Speichergröße
- Pro Nutzerdesign eine Page Table
- Tabelle zur Verwaltung freier Seiten

3 Virtualisierung

Tabellen

Virtuelle Adresse	Physische Adresse	gültig
0x00	pAddr 0	ja
0x01	pAddr 1	nein
...	...	...
2^{n-1}	pAddr 2^{n-1}	ja

User Page Table

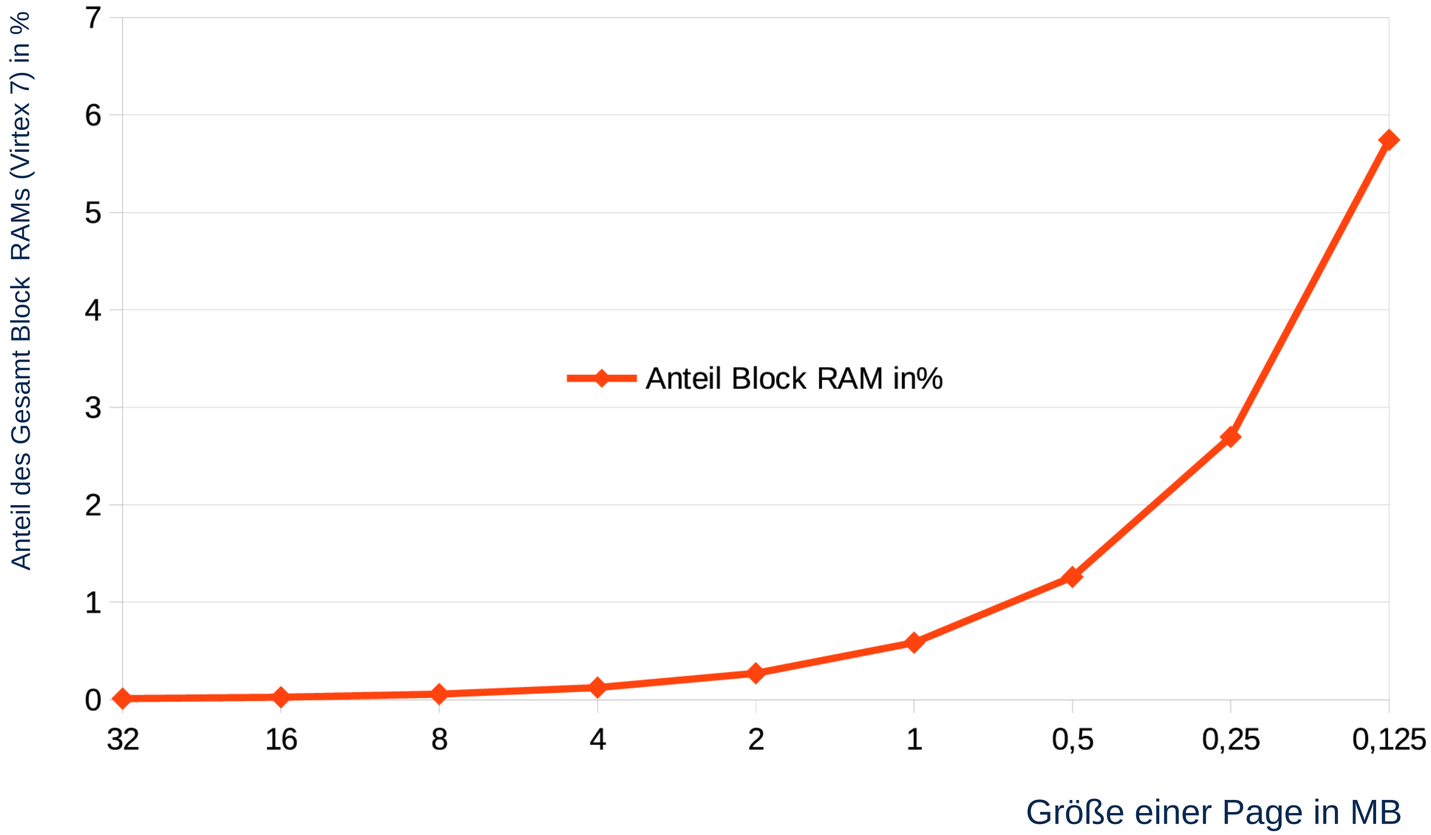
Adresse	benutzt
0x00	ja
0x01	ja
...	...
2^{n-1}	nein

Tabelle für freie Seiten

3 Virtualisierung

Größe der PageTables

- 4 User Page Tables: $4 \cdot (n+1) \cdot 2^n$ Bit
- Free Page Table: 2^n Bit
- Block RAM: 37 MBit
  (Virtex 7VX485T)
- n : Bits eines Page Table Eintrags



4 Konkurierender Zugriff

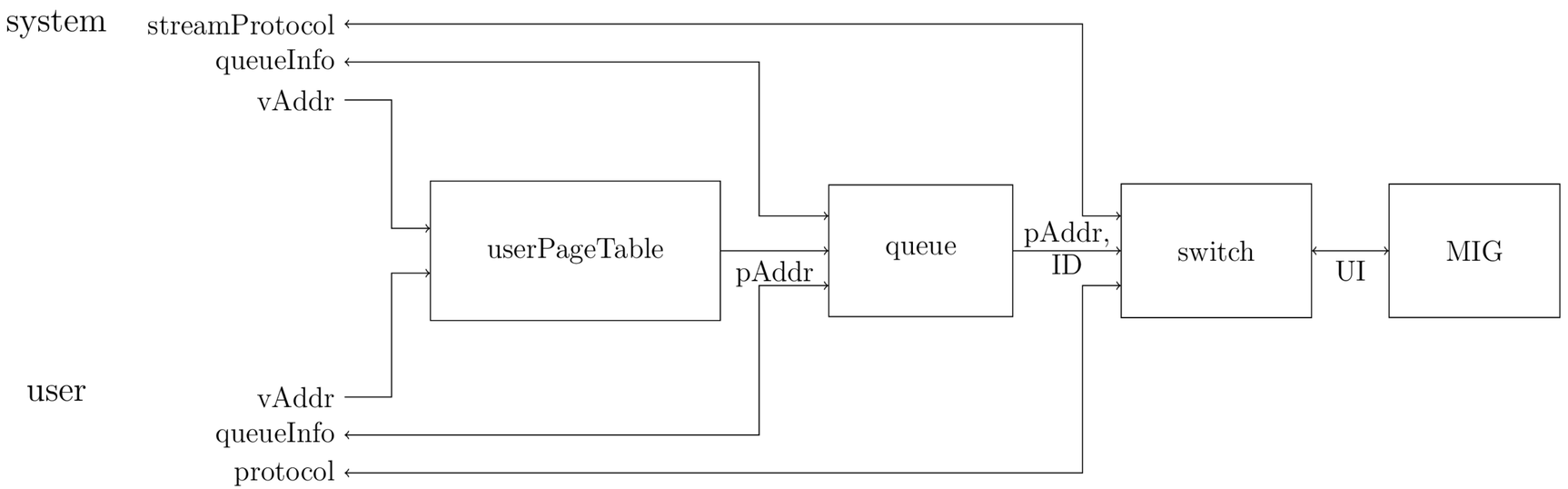
Arbitrierung

- Mehrere Nutzer
- Zeitgleicher Ressourcenzugriff
- Warteschlange
  - Voraussichtlich Round Robin/ FIFO
  - Evtl. Priorität für Systemzugriffe

5 Entwurf

Speicherzugriff

- 4 Page Tables
- Warteschlange
- Switch
- MIG

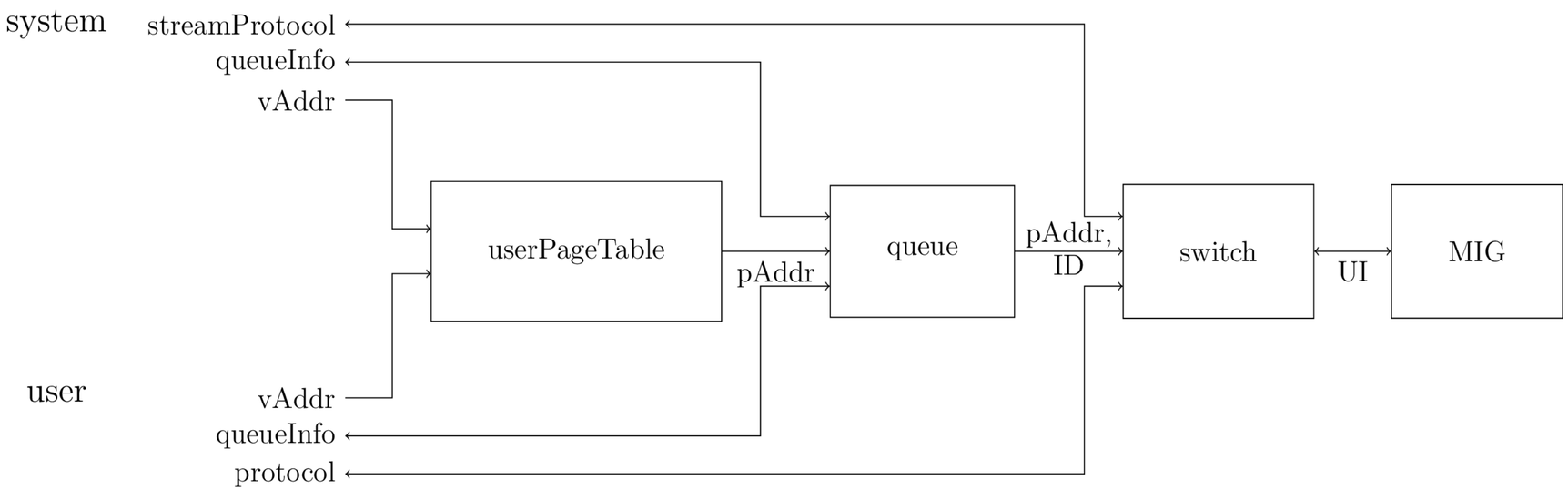


Konzept der Speicherzugriffs

5 Entwurf

Speicherzugriff

- (Stream)Protocol
  - Steuerleitung
  - Datenleitung
- QueueInfo
  - ID
  - req/ack

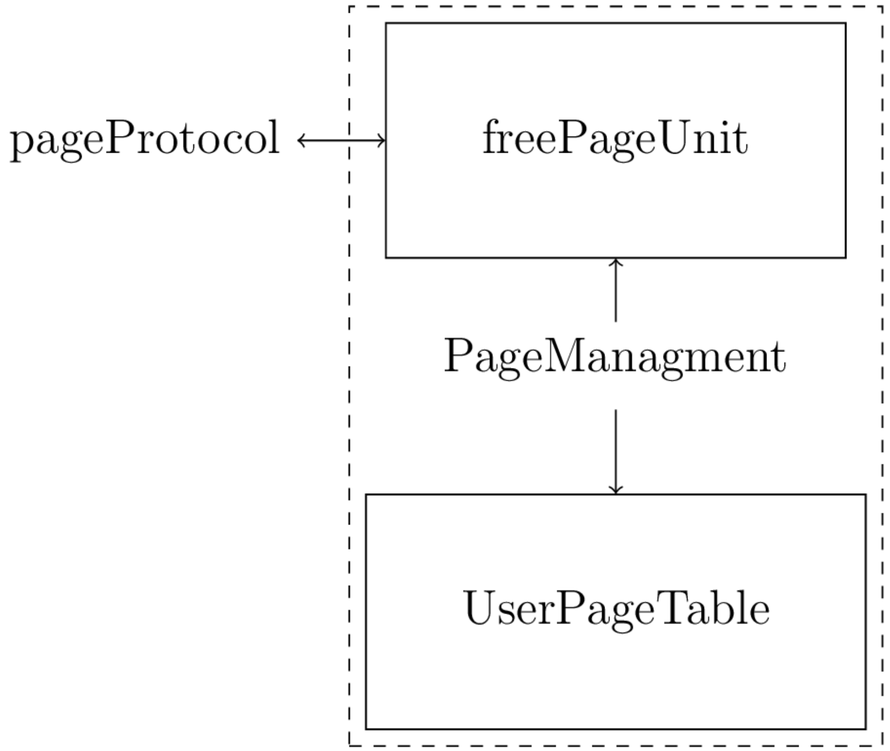


Konzept der Speicherzugriffs

5 Entwurf

Page Management

- Tabelle für freie Pages
- User Page Tables
- Verwaltung

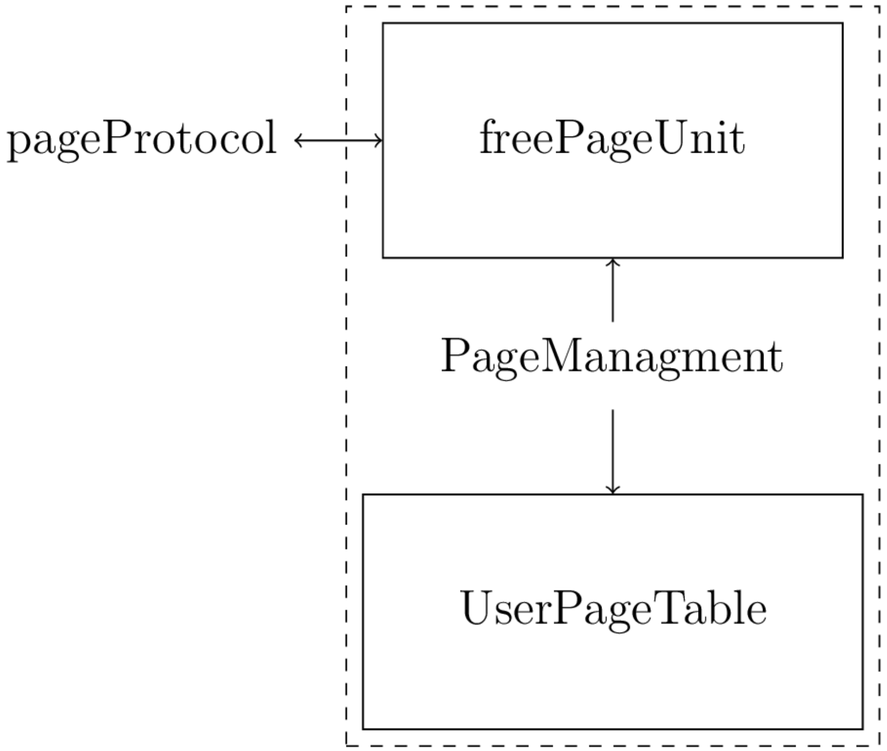


Konzept der Seitenverwaltung

5 Entwurf

Page Management

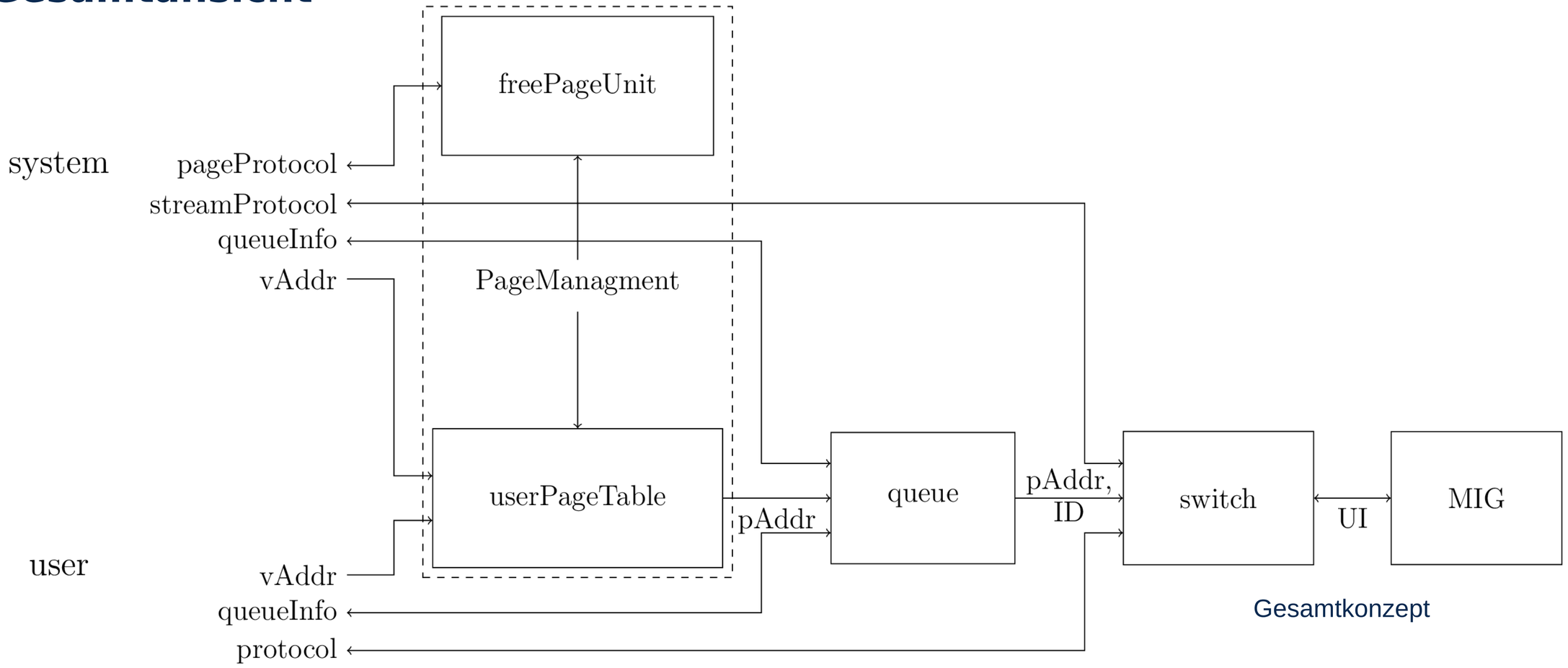
- PageProtocol
  - malloc, free
  - Virtuelle Adresse
  - full
- PageManagement
  - Modifizieren der User Page Table
  - Überprüfen: alter Eintrag gültig
  - Bei free Physische Adresse zurückgeben



Konzept der Seitenverwaltung

5 Entwurf

Gesamtansicht

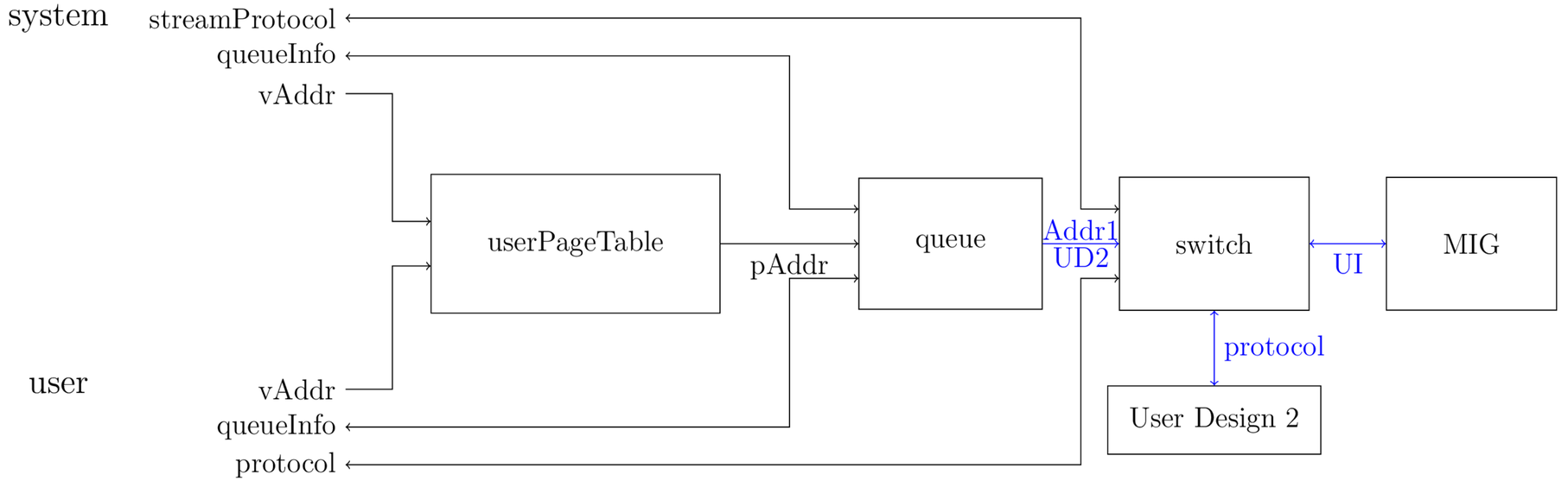


Gesamtkonzept

5 Entwurf

Szenario: Konkurrierender Speicherzugriff

- Nutzer 2 aktiv
- Nutzer 1 möchte auch zugreifen
  - Physische Adresse bestimmen
  - In Queue einreihen
- System-seitiger Zugriff von Nutzer 1 (danach)



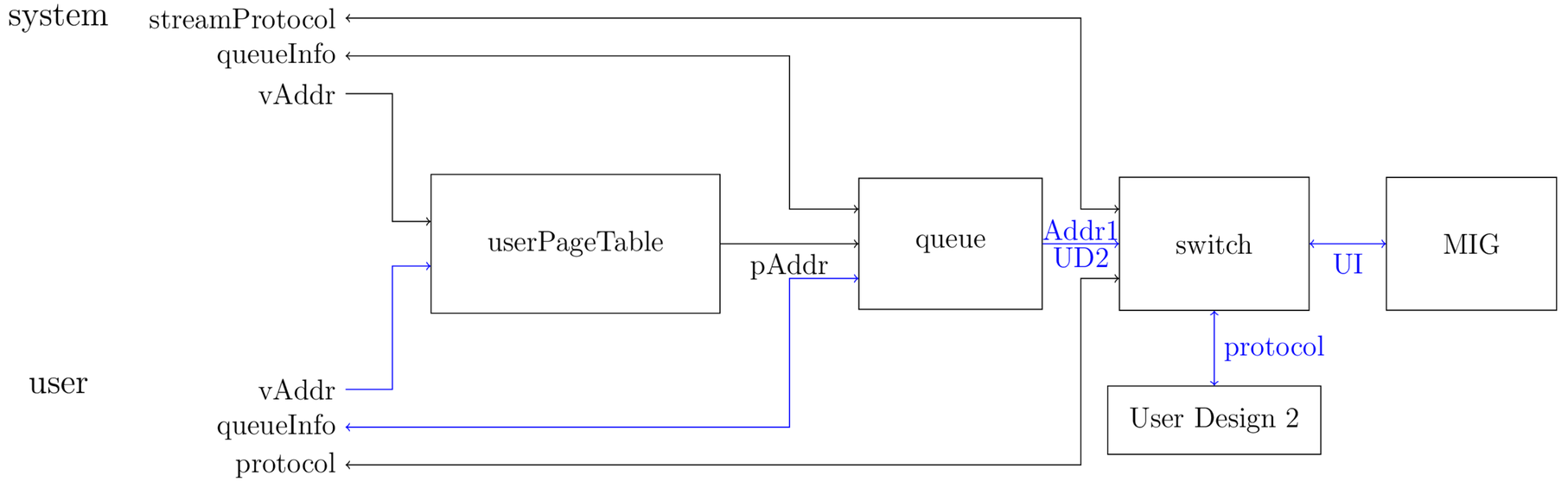
Szenario: Konkurrierender Speicherzugriff (Schritt 1)

Nutzerdesign 2 greift auf den Speicher zu

5 Entwurf

Szenario: Konkurrierender Speicherzugriff

- Nutzer 2 aktiv
- Nutzer 1 möchte auch zugreifen
  - Physische Adresse bestimmen
  - In Queue einreihen
- System-seitiger Zugriff von Nutzer 1 (danach)



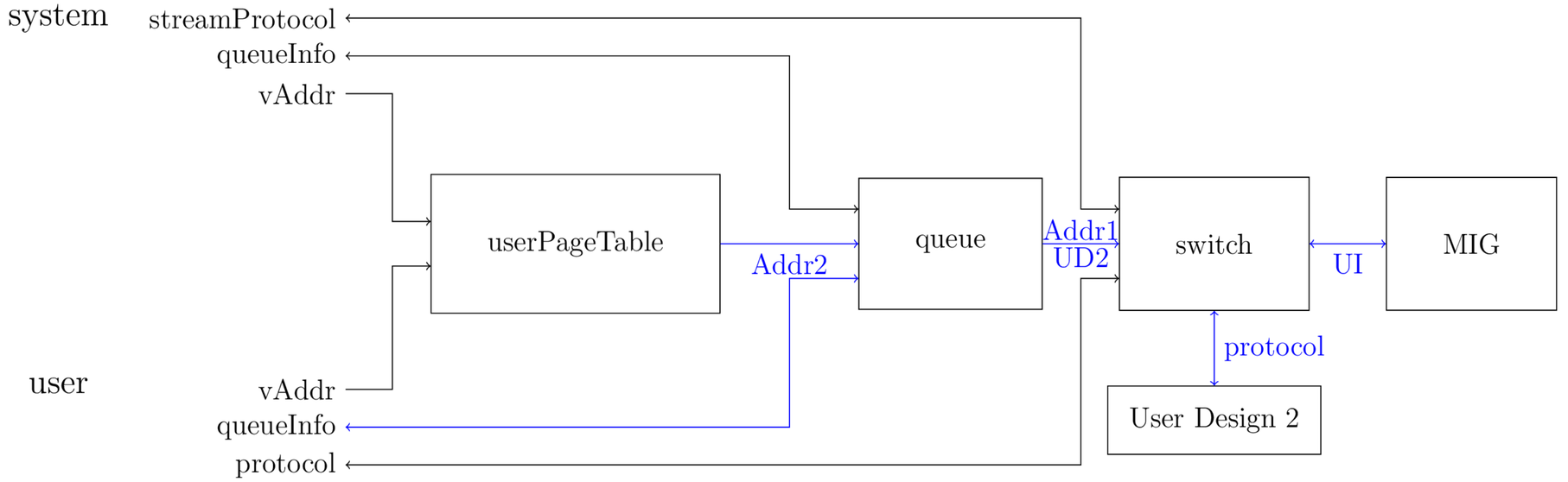
Szenario: Konkurrierender Speicherzugriff (Schritt 2)

Nutzerdesign 1 fordert auch Speicherzugriff und physische Adresse an

5 Entwurf

Szenario: Konkurrierender Speicherzugriff

- Nutzer 2 aktiv
- Nutzer 1 möchte auch zugreifen
  - Physische Adresse bestimmen
  - In Queue einreihen
- System-seitiger Zugriff von Nutzer 1 (danach)



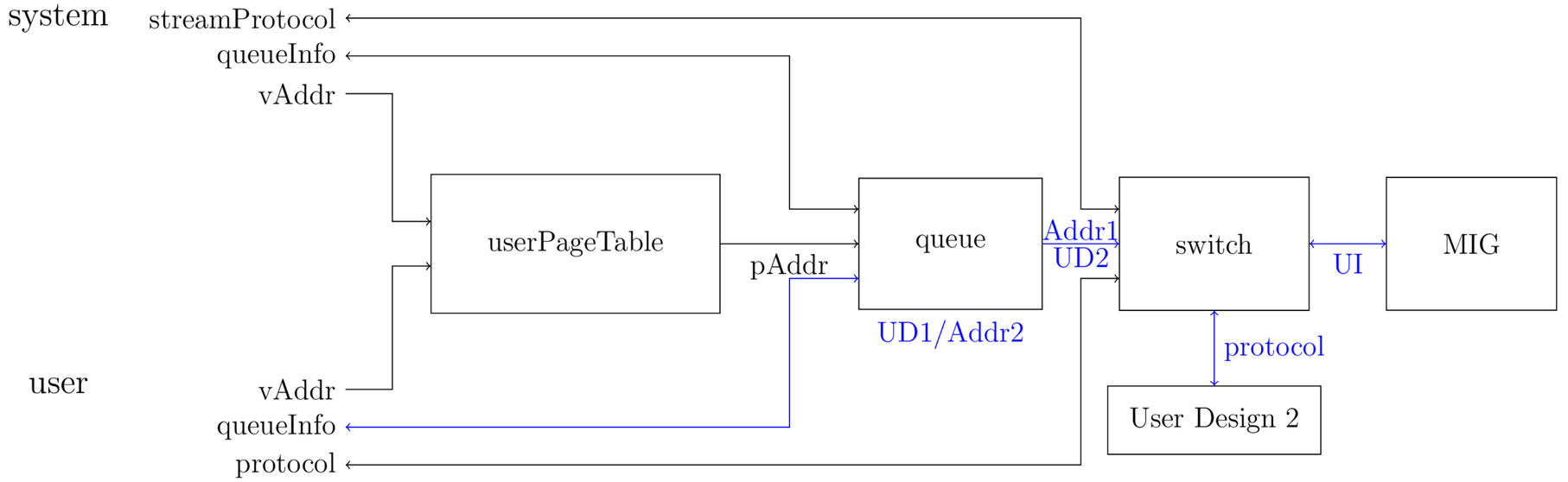
Szenario: Konkurrierender Speicherzugriff (Schritt 3)

Nutzerdesign 1 erhält die physische Adresse

5 Entwurf

Szenario: Konkurrierender Speicherzugriff

- Nutzer 2 aktiv
- Nutzer 1 möchte auch zugreifen
  - Physische Adresse bestimmen
  - In Queue einreihen
- System-seitiger Zugriff von Nutzer 1 (danach)



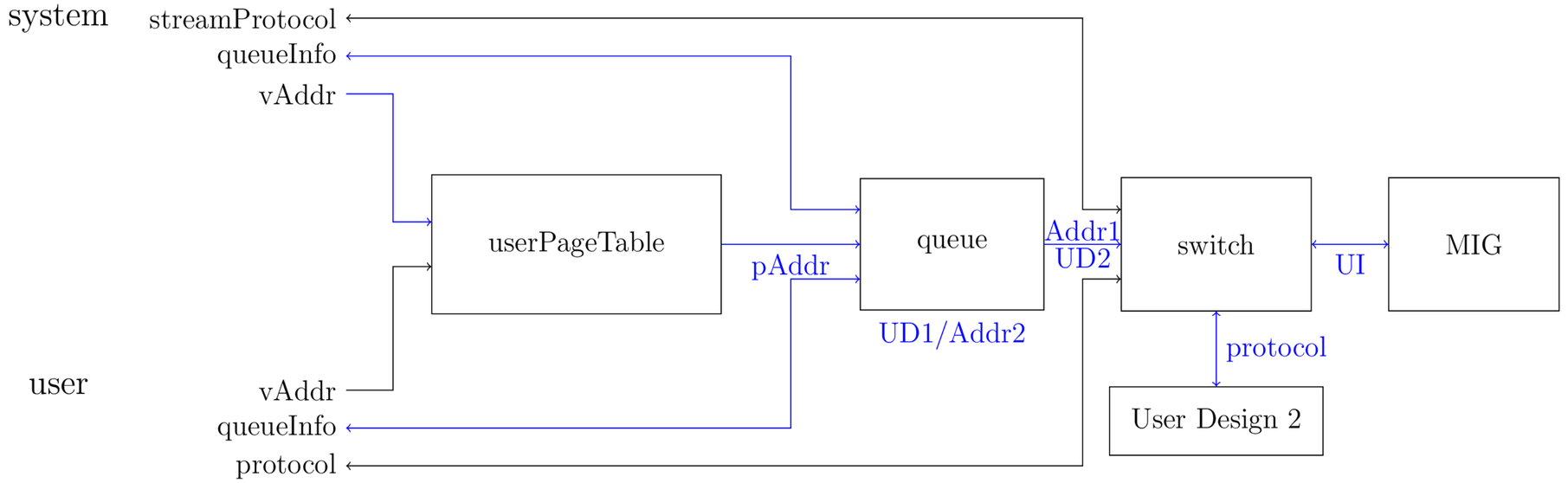
Szenario: Konkurrierender Speicherzugriff (Schritt 4)

Nutzerdesign 1 wird in die Warteschlange eingereiht.

5 Entwurf

Szenario: Konkurrierender Speicherzugriff

- Nutzer 2 aktiv
- Nutzer 1 möchte auch zugreifen
  - Physische Adresse bestimmen
  - In Queue einreihen
- System-seitiger Zugriff von Nutzer 1 (danach)



Szenario: Konkurrierender Speicherzugriff (Schritt 5)

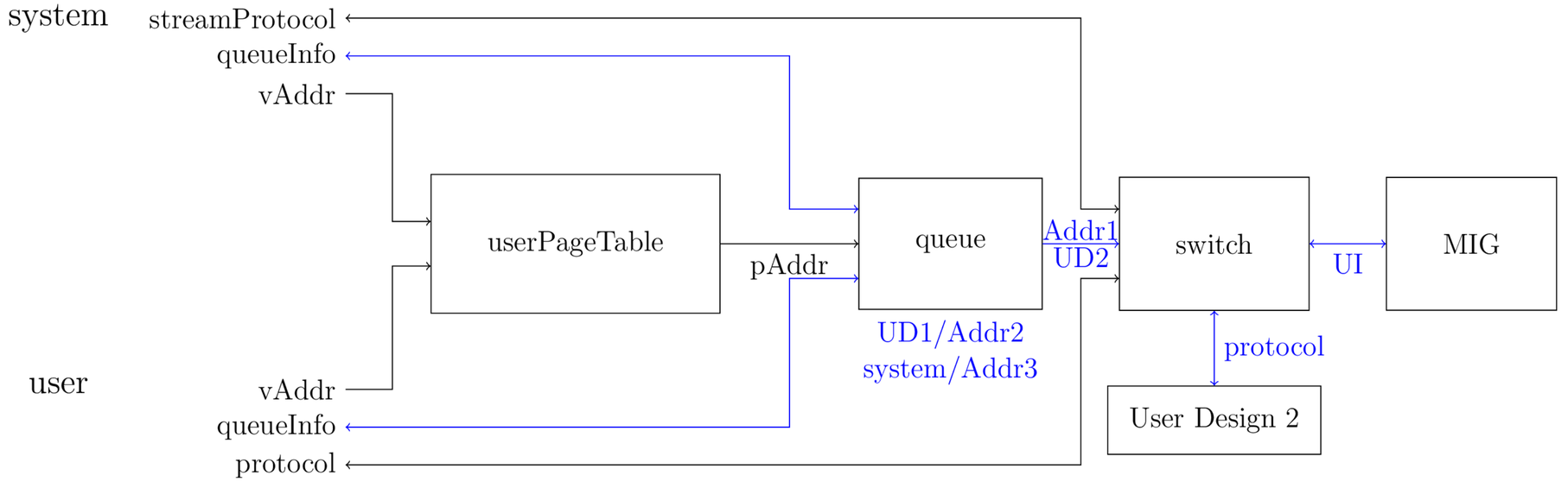
Nutzer1 möchte System-seitig auf dem Speicher zugreifen.

Das System erhält die Adresse aus der Page Table für Nutzer 1.

5 Entwurf

Szenario: Konkurrierender Speicherzugriff

- Nutzer 2 aktiv
- Nutzer 1 möchte auch zugreifen
  - Physische Adresse bestimmen
  - In Queue einreihen
- System-seitiger Zugriff von Nutzer 1 (danach)



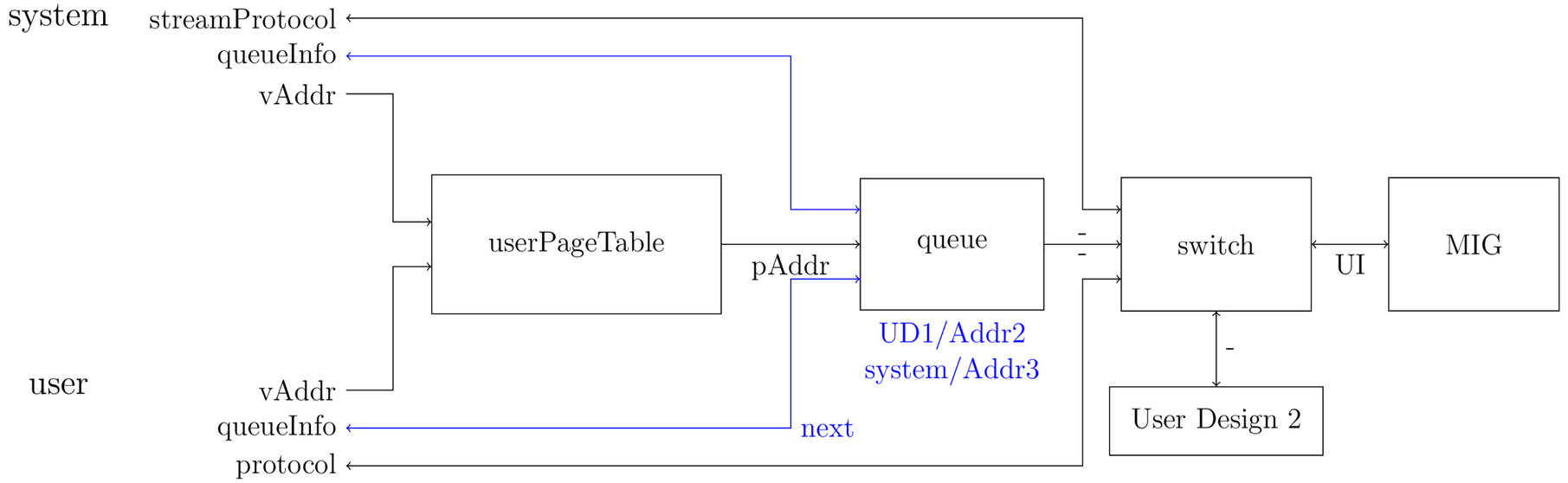
Szenario: Konkurrierender Speicherzugriff (Schritt 6)

Das System wird eingereiht.

5 Entwurf

Szenario: Konkurrierender Speicherzugriff

- Nutzer 2 aktiv
- Nutzer 1 möchte auch zugreifen
  - Physische Adresse bestimmen
  - In Queue einreihen
- System-seitiger Zugriff von Nutzer 1 (danach)



Szenario: Konkurrierender Speicherzugriff (Schritt 7)

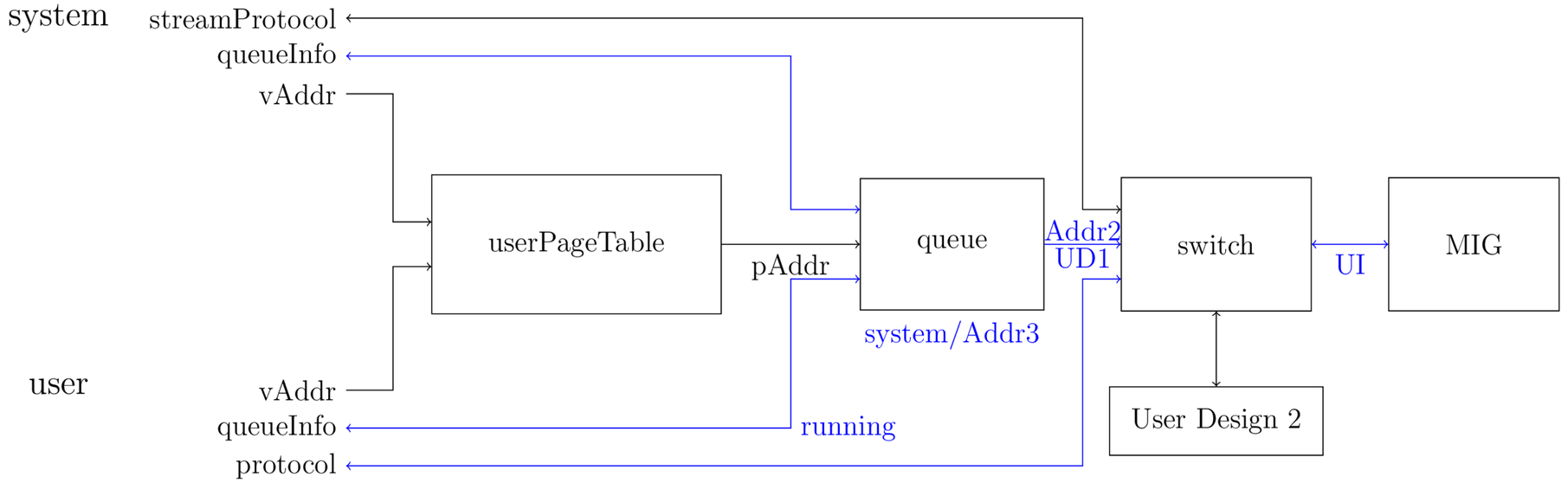
Nutzer Design 1 hat seinen Speicherzugriff beendet.

Der nächste Zugreifende wird aus der Warteschlange herausgesucht.

5 Entwurf

Szenario: Konkurrierender Speicherzugriff

- Nutzer 2 aktiv
- Nutzer 1 möchte auch zugreifen
  - Physische Adresse bestimmen
  - In Queue einreihen
- System-seitiger Zugriff von Nutzer 1 (danach)



Szenario: Konkurrierender Speicherzugriff (Schritt 8)

Nutzer 1 beginnt den Speicherzugriff.

6 Zusammenfassung

Aktueller Stand

- MIG
  - Testimplementierung zum UI
- Literatur
  - Virtualisierung und konkurrierende Zugriffe eingearbeitet
- Entwurfskonzept
- Implementierung der Komponenten
  - Page Management angefangen

6 Zusammenfassung

Ausblick

- Verfeinerung des Entwurfs
- Nutzerprotokoll festlegen
- Implementieren aller Einzelkomponenten
- Test der Einzelkomponenten
- Zusammenschalten der Einzelkomponenten
- Testen des Gesamtentwurfs
- Messen der Performance

7 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. Virtex-7 T and XT FPGAs Data Sheet: DC and AC Switching Characteristics

[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