2. Der *ParaNut*-Prozessor

"Parallel and more than just another CPU core"

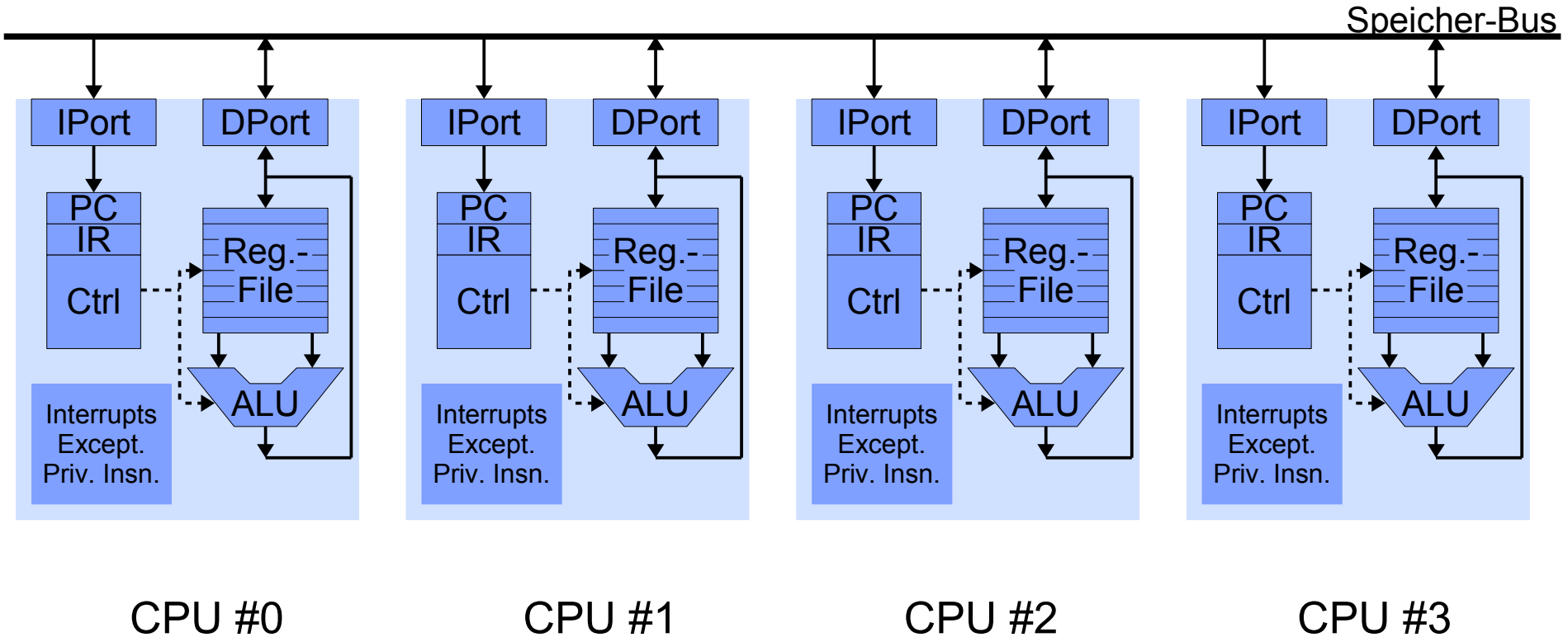
- **Neuer, konfigurierbarer Prozessor**
- **Parallelität auf Daten- (SIMD) und Thread-Ebene**
  - neues Konzept mit guter Hochsprachen-Unterstützung
- **Hohe Skalierbarkeit mit *einer* Architektur**
  - je nach Konfiguration: klein oder schnell
- **Open Source**
- **Praxistauglichkeit durch Orientierung an existierender Befehlssatzarchitektur**
  - ***OpenRISC 1000*** (auch möglich: MIPS, SPARC, ARM, ...)
  - Toolkette: ***GCC***
  - Betriebsumgebung: ***newlib, Linux***

Motivation für das Projekt

- **Bedarf an skalierbarem, parallelem Soft-Core-Prozessor (z. B. im *Triokulus*-Projekt ...)**
  - *LEON, OpenSPARC*: hoher Flächenbedarf
  - *Microblaze, OpenRISC*: keine/unzureichende Multi-Core-Unterstützung
- **Praxiseinsatz von FPGA-basierten Ein-Chip-Systemen**
  - technisch oft sinnvoll, aber Lizenzkosten für proprietäre Soft-Core-Prozessoren schrecken ab
- **Einsatz in der Lehre**
  - FPGA-Entwicklung, Compiler, Betriebssysteme, HW/SW-Codesign
- **Reputation der HSA durch Open-Source-Projekt**

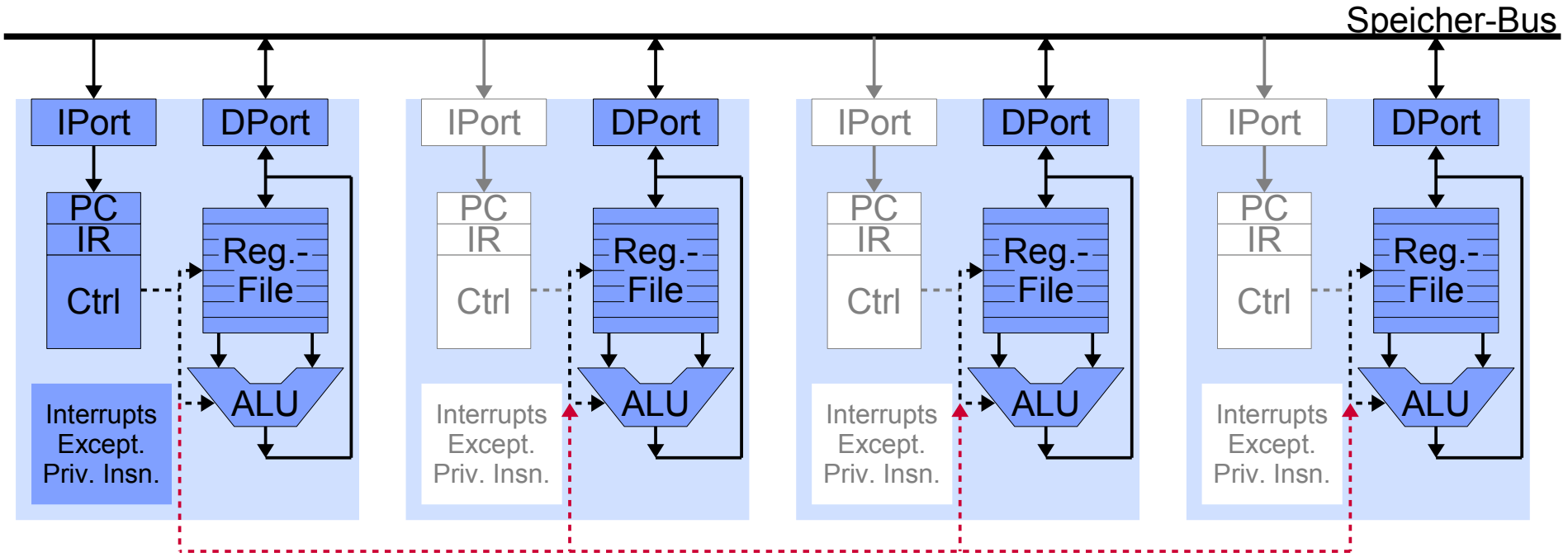
Grundidee

- **Was passiert, wenn man bei einem Multi-Core die Steuerung zusammenschaltet?**



Grundidee

- **Was passiert, wenn man bei einem Multi-Core die Steuerung zusammenschaltet?**



- **Hardware:** Entspricht Vektor-(SIMD-)Prozessor
- **Software:** Verhalten wie Multi-Core außer bei *bedingten Sprüngen*

Vektorisierung in Hochsprache

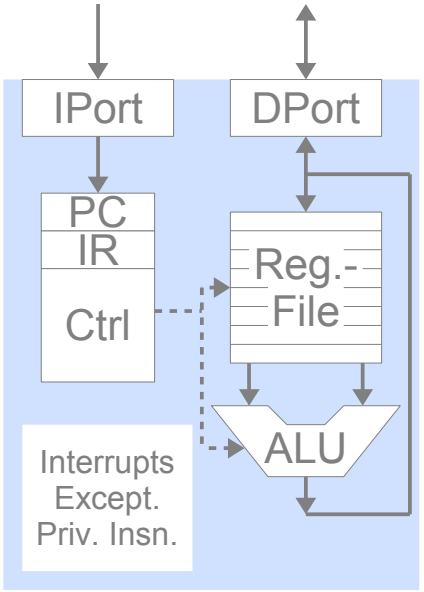
- **Viele andere SIMD-Erweiterungen (*MMX, SSEn*) sind nur in Assembler sinnvoll nutzbar**
  - Spezialbefehle (z.B. "add with saturation") nicht von Compiler nutzbar
  - fehlende Flexibilität (z.B. bei Speicherzugriffen) schränkt Nutzung ein

- ***ParaNut*: Hochsprache, ähnlich Multi-Threading**

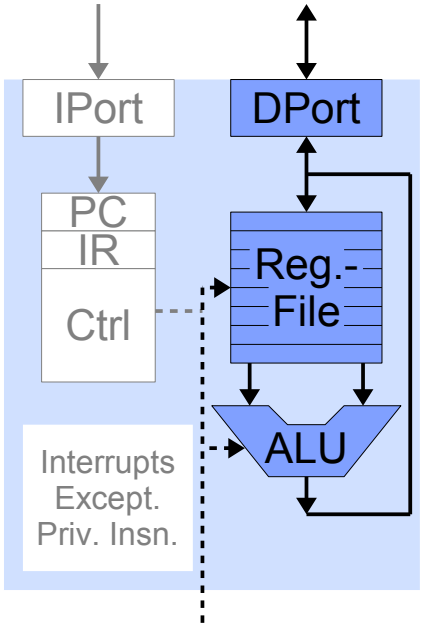
```
int a[1000], b[1000], s[1000];
int n, id;
...

id = pn_begin_linked (4); // returns CPU id (0..3)
for (n = 0; n < 1000; n += 4)
  s[n + id] = a[n + id] + b[n + id];
pn_end_linked ();
```

Modi und Ausbaustufen (*Capabilities*)

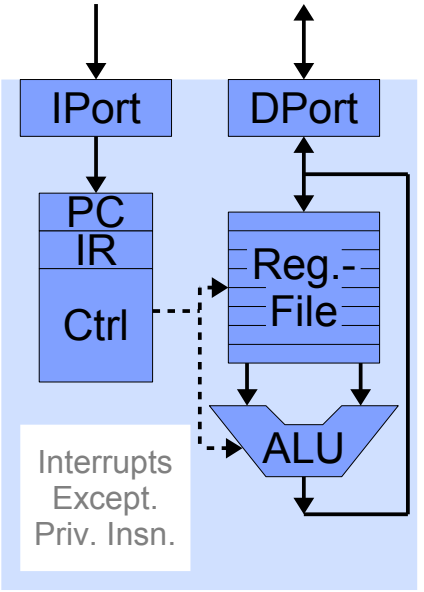
Mode 0
(Inactive)



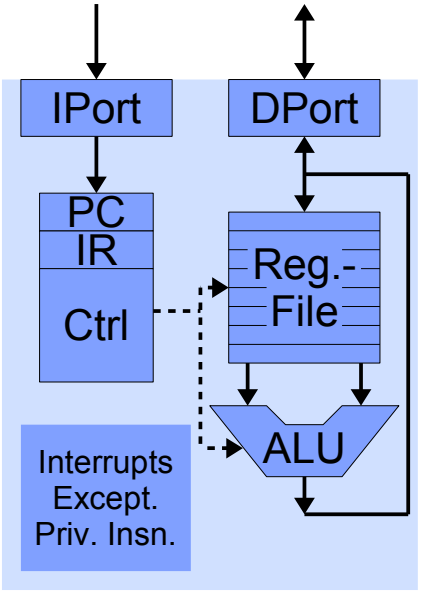
Mode 1
(Linked / Vector)



Mode 2
(Thread)

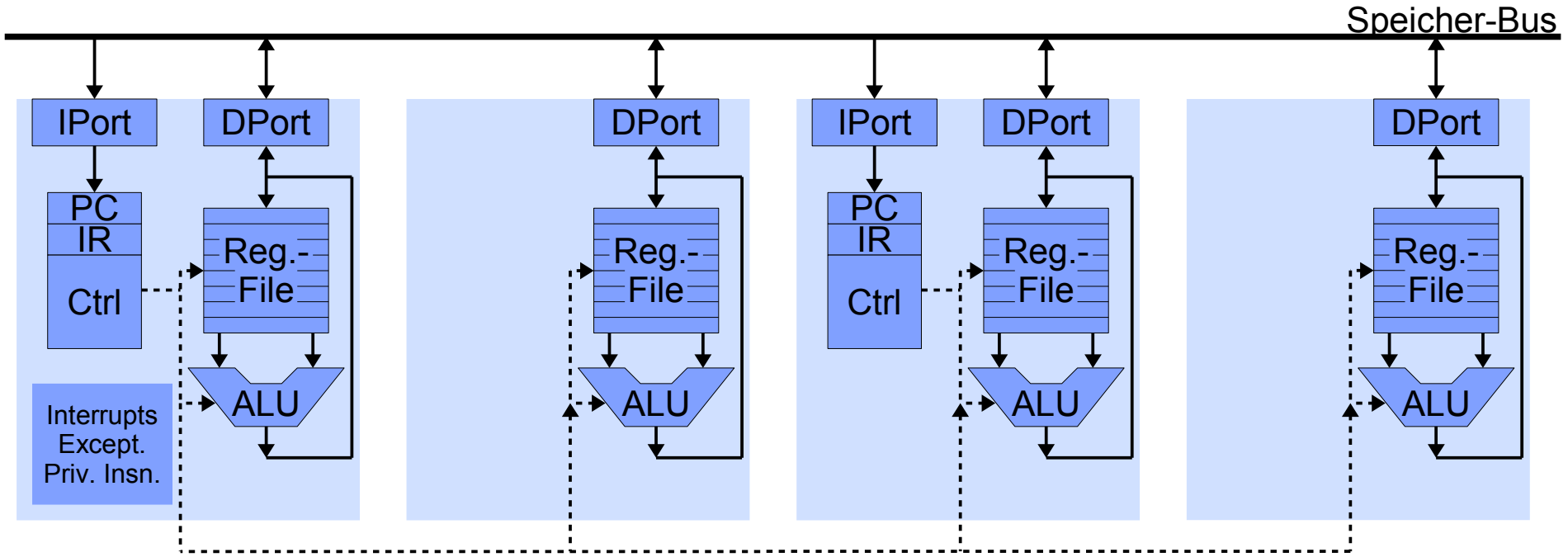


Mode 3
(Full)



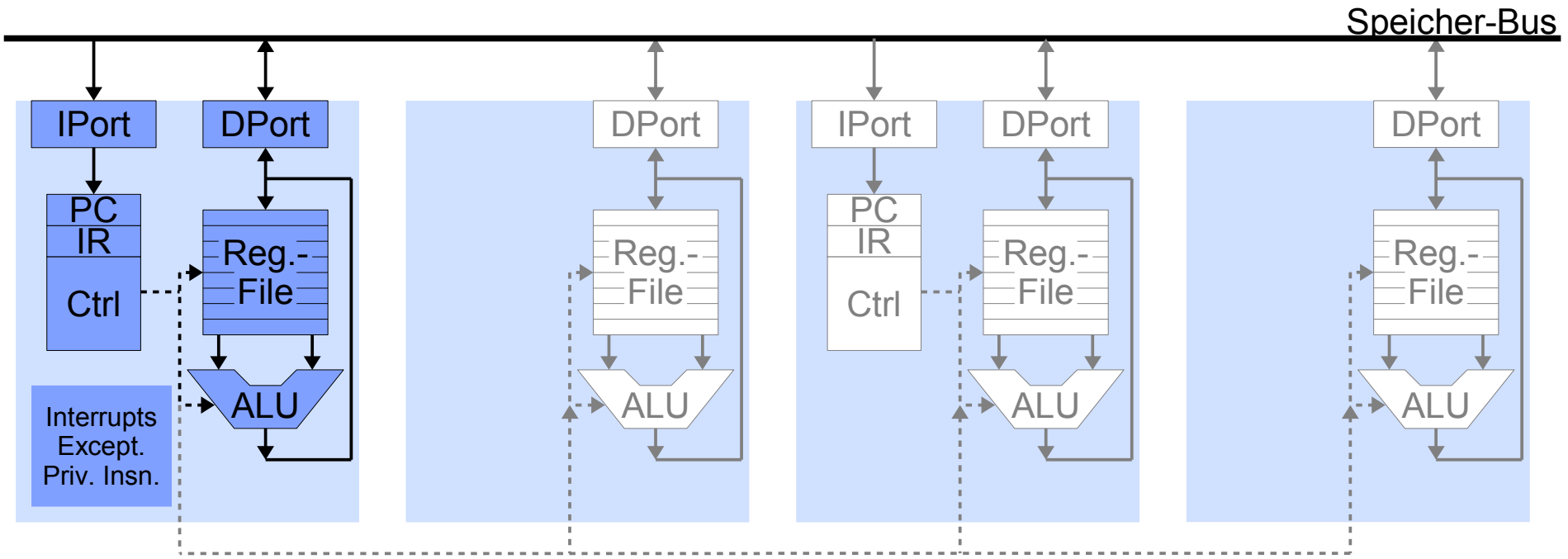
- Je nach Bedarf müssen nicht alle Modi unterstützt werden.
- ***Capability N*** : Alle Modi <= *N* werden unterstützt

Beispiel: *ParaNut* für 2 Threads und 4-fach-Vektorisierung



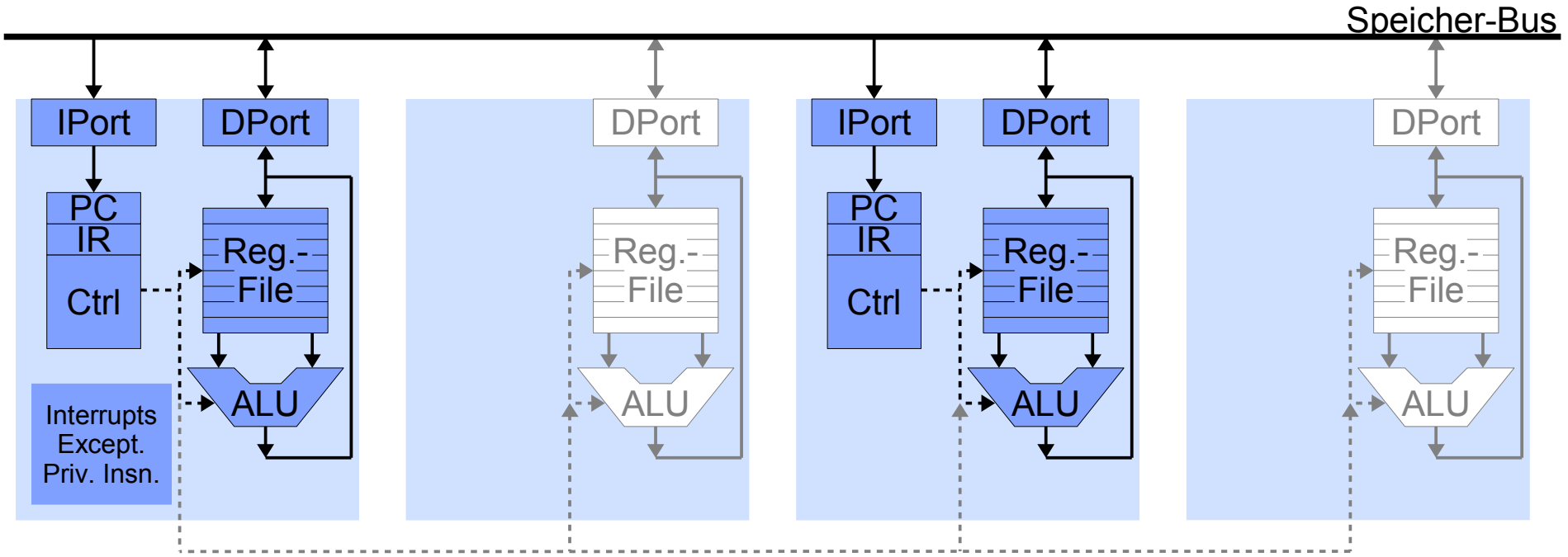
Capability: 3 1 2 1

Ein Thread, keine Vektorisierung



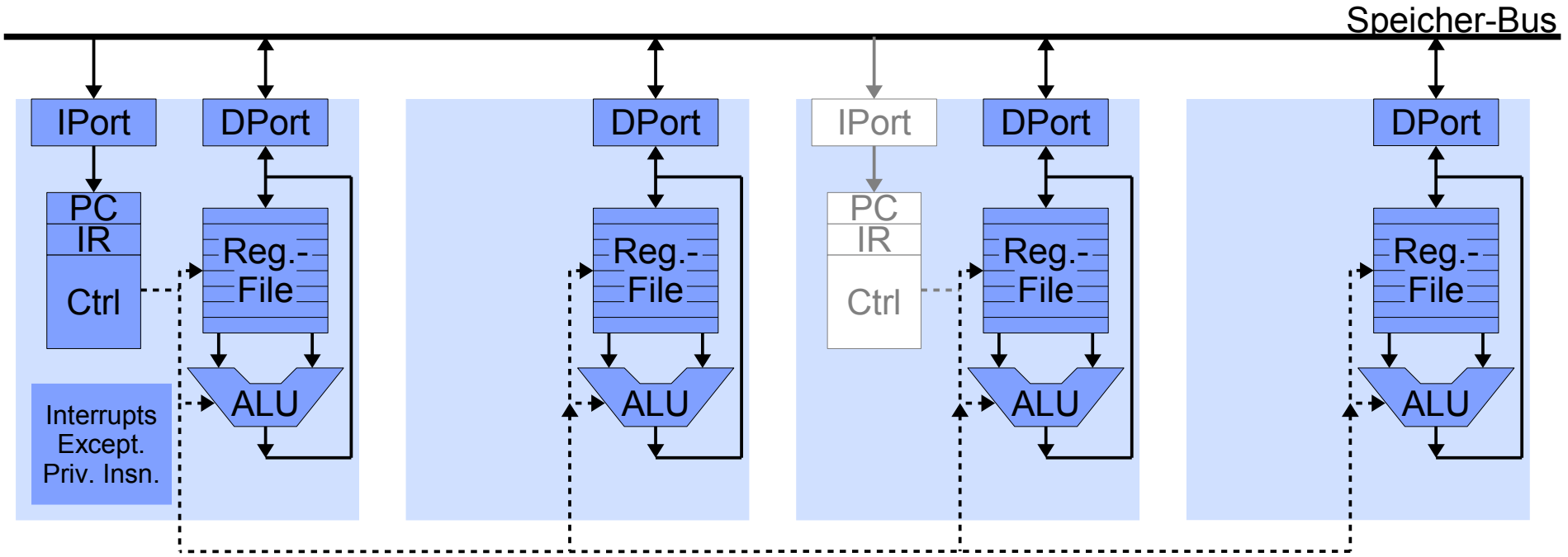
Capability:	**3**	**1**	**2**	**1**
Mode:	**3**	**0**	**0**	**0**

Zwei Threads



Capability:	**3**	**1**	**2**	**1**
Mode:	**3**	**0**	**2**	**0**

4-fach-Vektorrechner



Capability:	**3**	**1**	**2**	**1**
Mode:	**3**	**1**	**1**	**1**

Stand des Projektes

- **Top-Down-Entwurf mit *SystemC***
- **Referenz-Handbuch: "lebendes Dokument"**
- **Stand der Implementierung:**
  - **Simulations-Framework**
    - lädt ELF-Dateien und simuliert System
  - **Speicher-Modul (auf RT-Ebene)**
    - beinhaltet u.a. Cache & Synchronisationsmechanismen
    - hohe Effizienz (z.B. bei Vektor-Zugriffen) durch Banking und gespiegelte Tags
  - **Grundmodell des Prozessors (Verhaltens-Ebene, taktgenau)**
    - *OpenRISC*-Basisbefehlssatz (*ParaNut*-Erweiterungen in Arbeit)

-> Erste C-Programme erfolgreich compiliert und im Simulator ausgeführt,
aber noch viel zu tun ...

Nächste Schritte

- **Komplettierung des *SystemC*-Modells**
- **Implementierung in VHDL, Test auf realer Hardware**
- **Software-Bibliotheken**
  - *libparanut:* Support-Bibliothek (Modi steuern, Synchronisations-Primitive, ...)
  - *Pthreads* & *OpenMP*
  - Anpassung des (OpenRISC-)Linux-Kernels
- **Hardware-Erweiterungen (z.B. MMU)**

Teilaspekte können von Studenten (Projektarbeiten, Abschlussarbeiten) bearbeitet werden!