ISE SRA

Leibniz Universität Hannover

InterSloth: Global Hardware-Based Scheduling in a MultiCore-RTOS on RISC-V

Malte Bargholz, Matthias Wolf, Daniel Lohmann

2018-10-18

Prioritätsgesteuertes Echtzeitscheduling

Prioritätsgesteuertes Echtzeitscheduling

- Schedulingentscheidung nur anhand von festen Prioritäten (FP)

Prioritätsgesteuertes Echtzeitscheduling

- Schedulingentscheidung nur anhand von festen Prioritäten (FP)
- Partioniertes FP-Scheduling
  - Erweitertes Bin-Packing Problem $\Rightarrow$ NP-Schwer [Bra11]
  - Auslastungsverlust durch Approximationsalgorithmus
  - Unflexibel für veränderte Echtzeitparameter

Prioritätsgesteuertes Echtzeitscheduling

- Schedulingentscheidung nur anhand von festen Prioritäten (FP)
- Partioniertes FP-Scheduling
  - Erweitertes Bin-Packing Problem $\Rightarrow$ NP-Schwer [Bra11]
  - Auslastungsverlust durch Approximationsalgorithmus
  - Unflexibel für veränderte Echtzeitparameter
- Globales FP-Scheduling
  - Kein Auslastungsverlust durch Algorithmus
  - Höhere Adaptivität an Umweltänderungen
  - Hohe Overheads durch geteilte Daten- und Hardwarestrukturen

Prioritätsgesteuertes Echtzeitscheduling

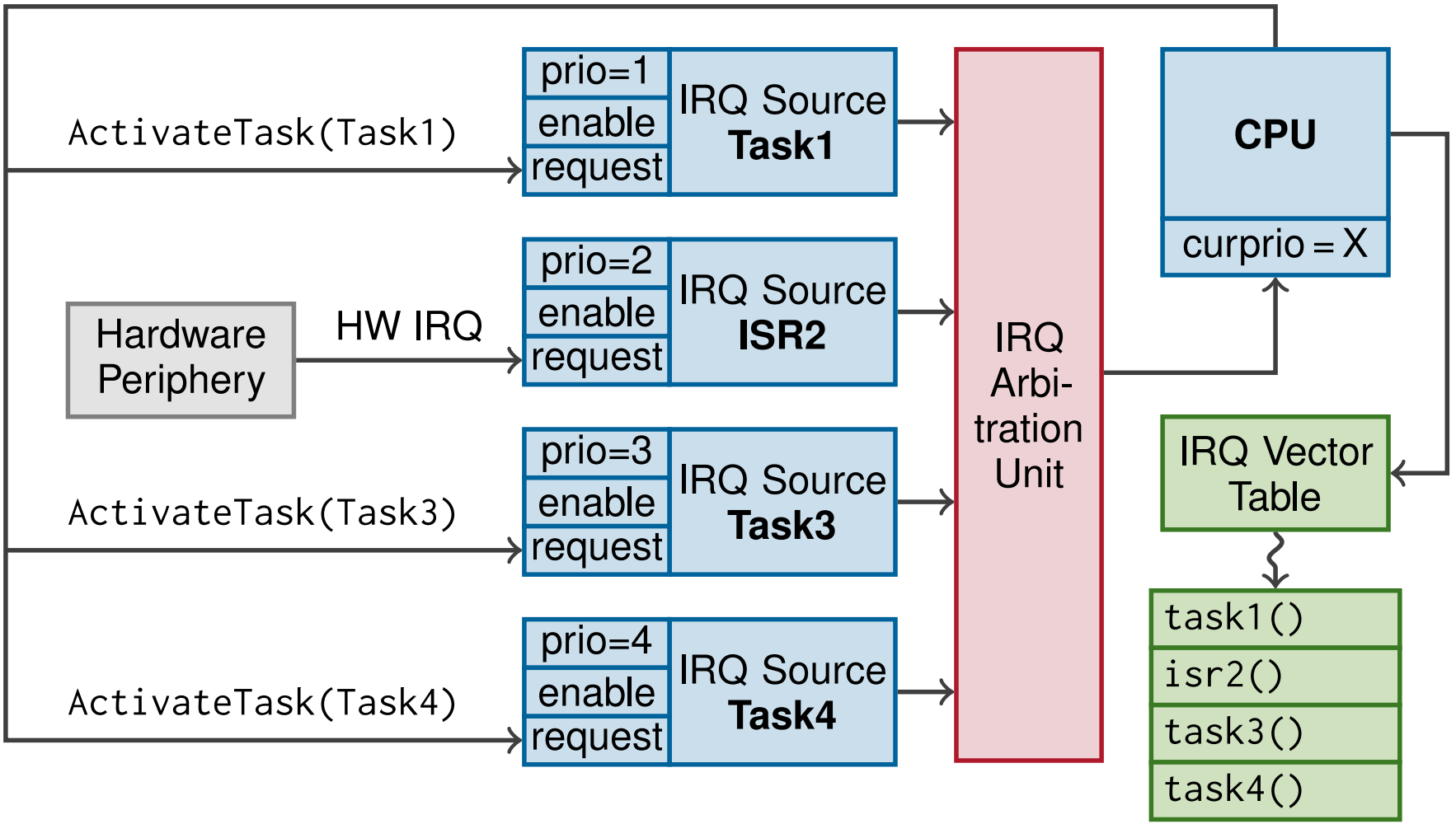
- Schedulingentscheidung nur anhand von festen Prioritäten (FP)
- Partioniertes FP-Scheduling
  - Erweitertes Bin-Packing Problem $\Rightarrow$ NP-Schwer [Bra11]
  - Auslastungsverlust durch Approximationsalgorithmus
  - Unflexibel für veränderte Echtzeitparameter
- Globales FP-Scheduling
  - Kein Auslastungsverlust durch Algorithmus
  - Höhere Adaptivität an Umweltänderungen
  - Hohe Overheads durch geteilte Daten- und Hardwarestrukturen

Ziel

Vermeidung von Synchronisierungsoverhead im Scheduler

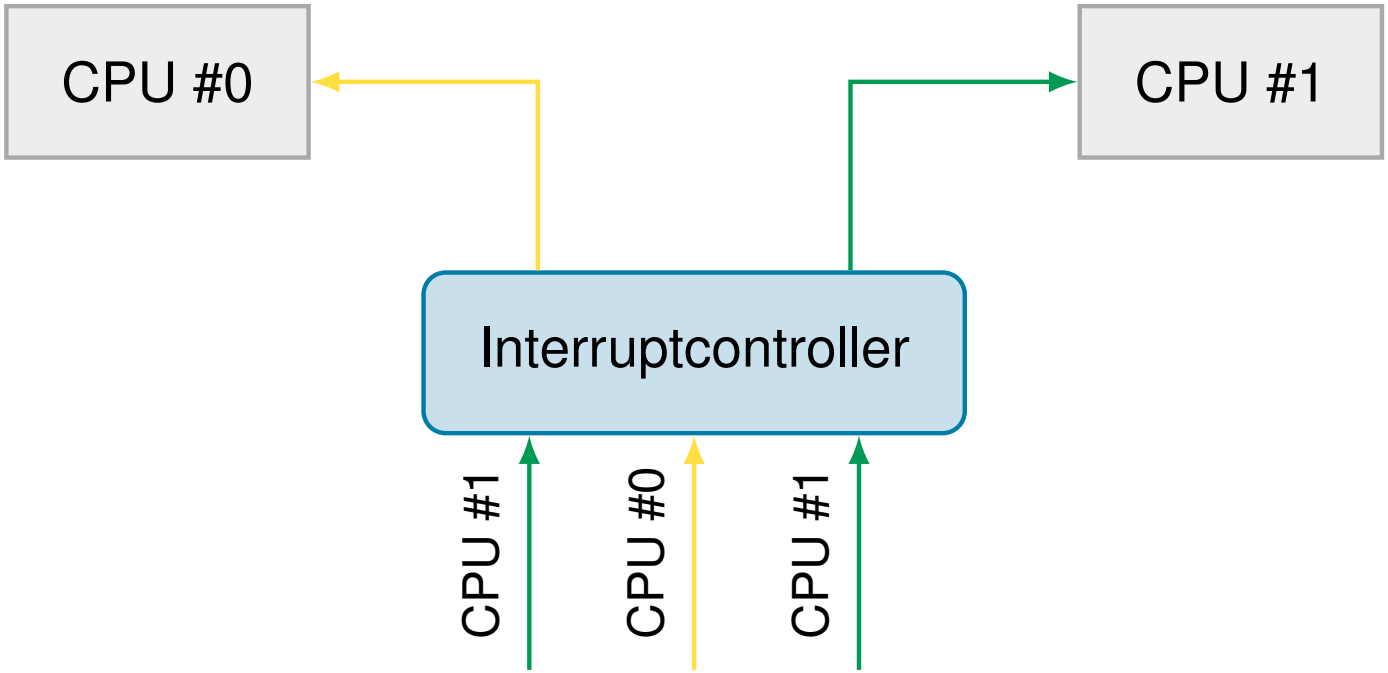
Sloth: Threads as Interrupts [Hof+09]

- Grundidee: Task ≡ ISR



MultiSloth [Mül+14]

- Multi-Core Sloth **aber:** partionierter Ansatz
- Tasks sind permanent an Prozessorkerne gebunden
- Beschränkung durch verfügbare Interrupt Controller



Motivation

MIRQ-V

InterSloth

Evaluation

Zusammenfassung

MIRQ-V

- Verfügbare Interruptcontroller (ARM, Infineon) priorisieren pro CPU

MIRQ-V

- Verfügbare Interruptcontroller (ARM, Infineon) priorisieren pro CPU
- MIRQ-V: Priority-Obedient Multicore Interrupt Controller [Bew16]
  - Plattformunabhängiger Interruptcontroller
  - Unterstützt globale Priorisierung und Interruptmigration
  - HDL: Chisel $\Rightarrow$ kompatibel mit Zielplatform RISC-V

MIRQ-V

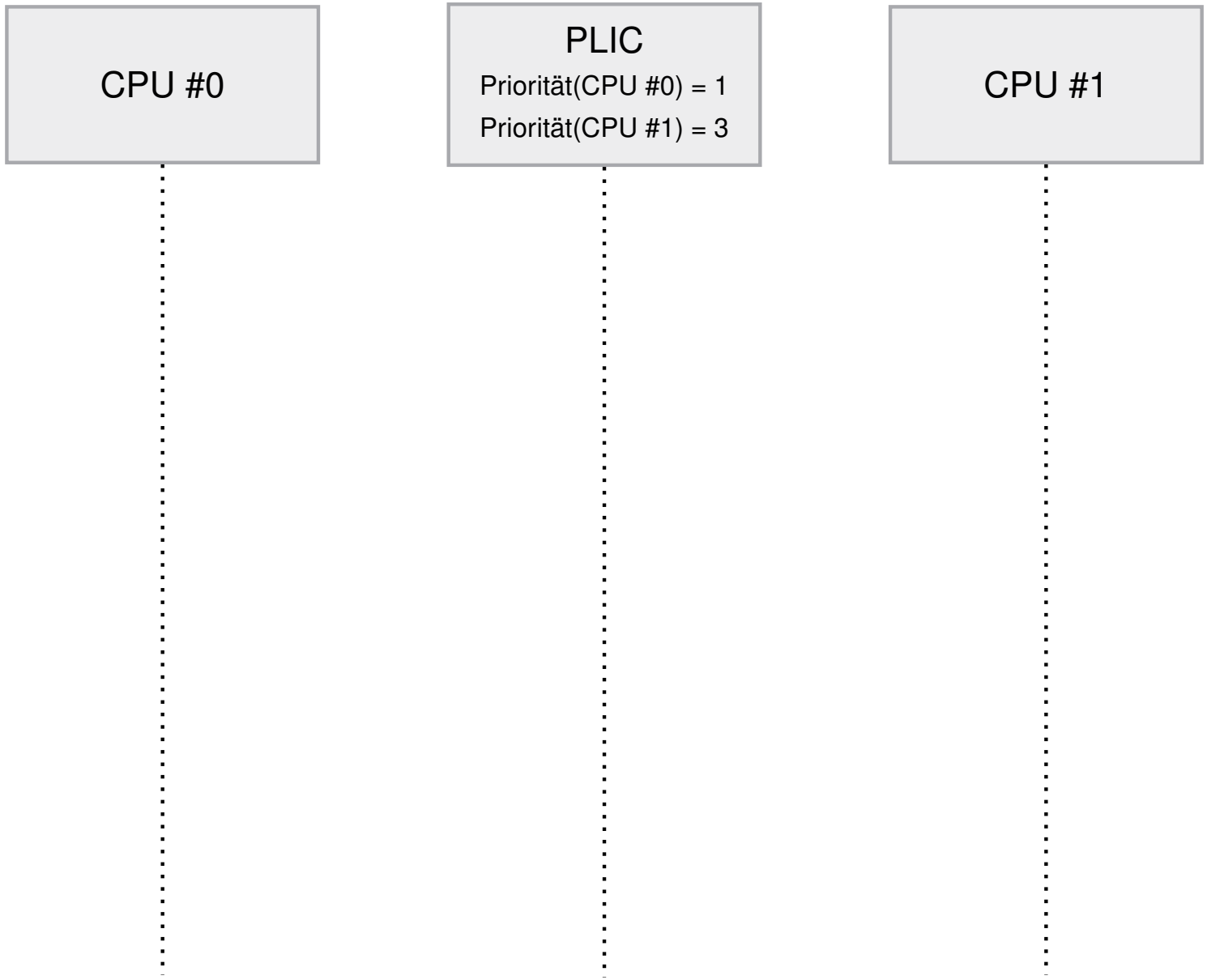
- Verfügbare Interruptcontroller (ARM, Infineon) priorisieren pro CPU
- MIRQ-V: Priority-Obedient Multicore Interrupt Controller [Bew16]
  - Plattformunabhängiger Interruptcontroller
  - Unterstützt globale Priorisierung und Interruptmigration
  - HDL: Chisel $\Rightarrow$ kompatibel mit Zielplatform RISC-V
- Anpassung von einer RISC-V-Referenzimplementierung (`rocket-chip`) auf MIRQ-V Funktionalität

MIRQ-V

- Verfügbare Interruptcontroller (ARM, Infineon) priorisieren pro CPU
- MIRQ-V: Priority-Obedient Multicore Interrupt Controller [Bew16]
  - Plattformunabhängiger Interruptcontroller
  - Unterstützt globale Priorisierung und Interruptmigration
  - HDL: Chisel $\Rightarrow$ kompatibel mit Zielplatform RISC-V
- Anpassung von einer RISC-V-Referenzimplementierung (`rocket-chip`) auf MIRQ-V Funktionalität
- Zielplatform RISC-V besitzt zwei Interruptcontroller
  - *Platform-Level Interrupt Controller* (PLIC): globale Interrupts
  - *Core-Local Interruptor* (CLINT): lokale Interrupts

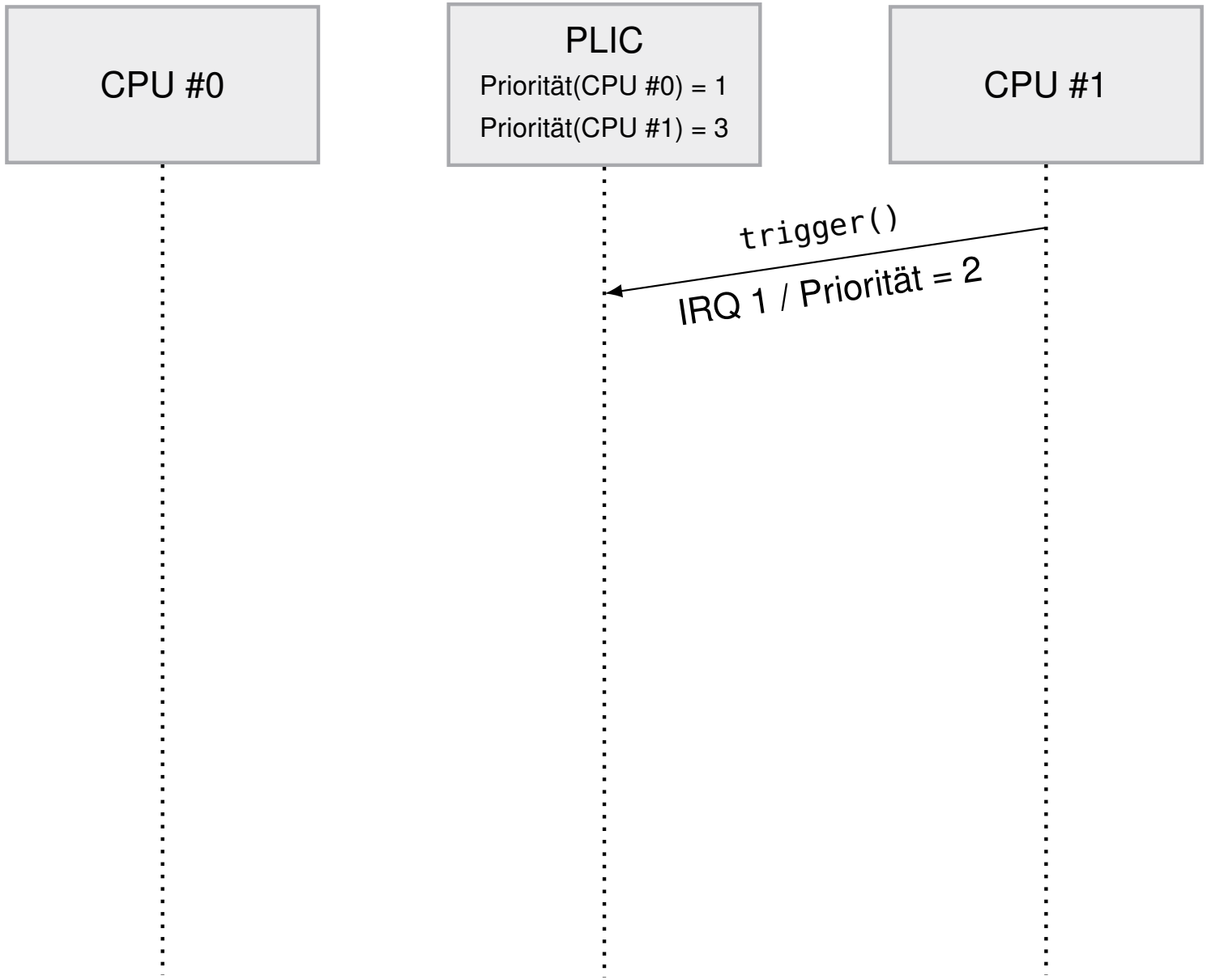
MIRQ-V in RISC-V

Beispiel Interruptverlauf



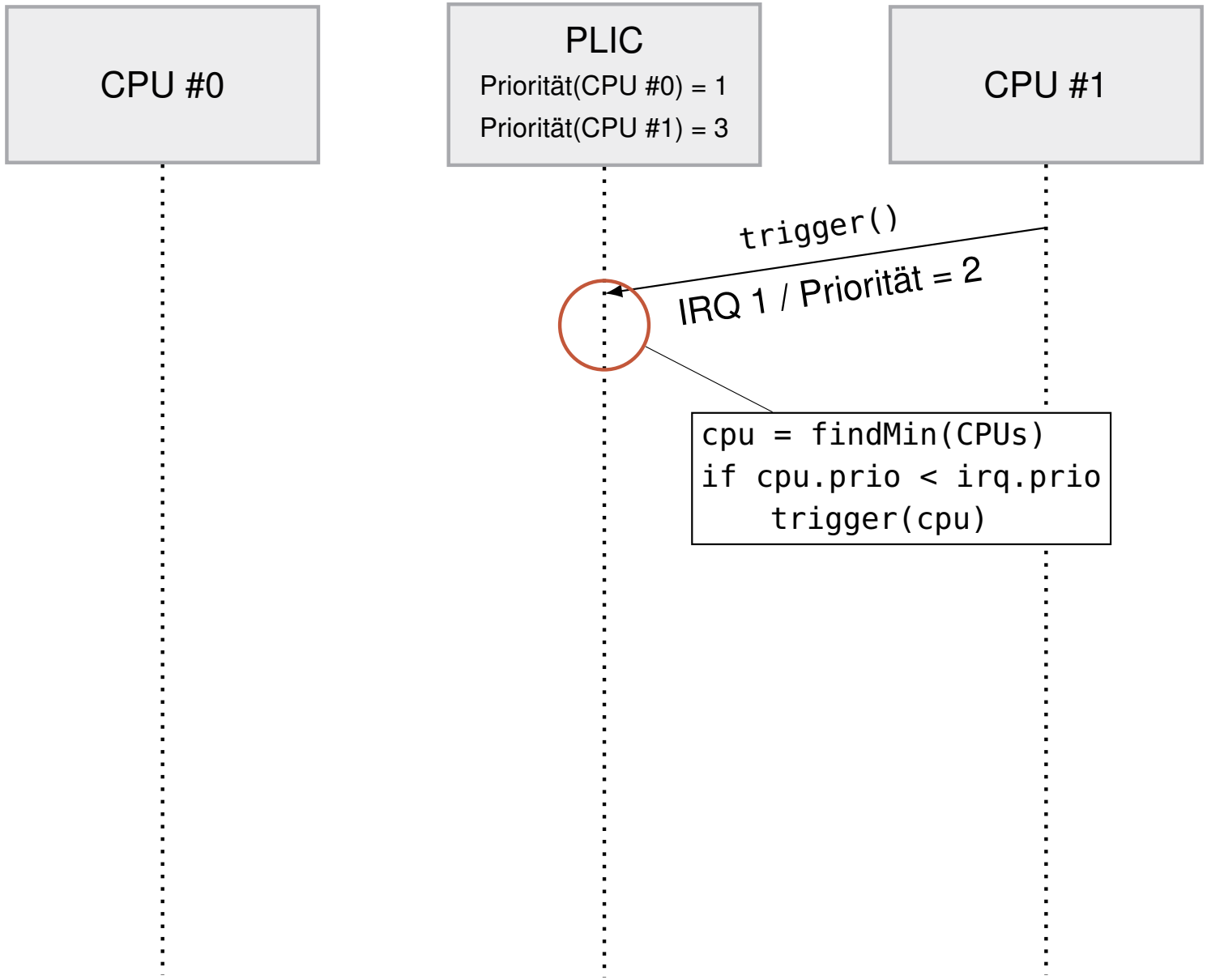
MIRQ-V in RISC-V

Beispiel Interruptverlauf



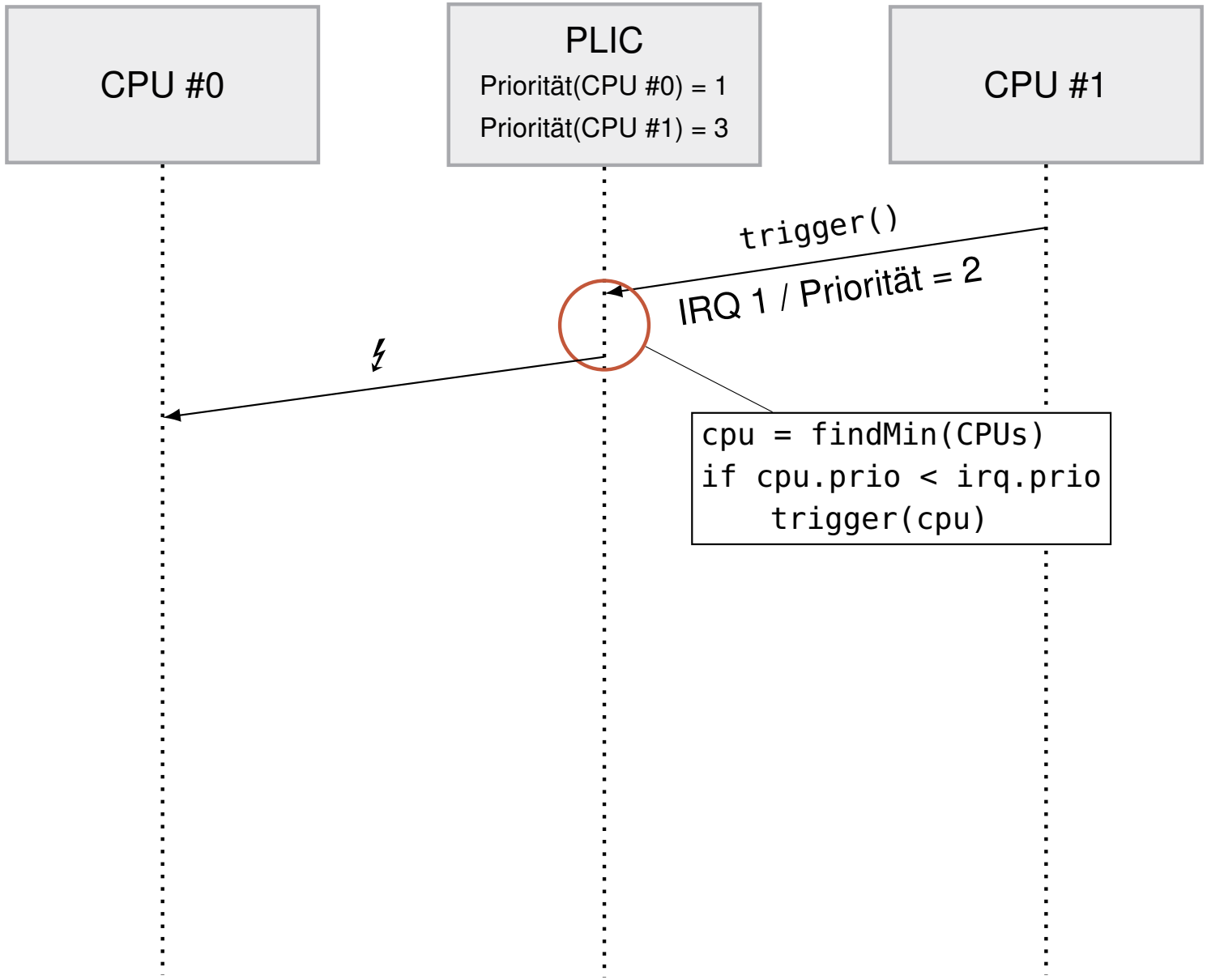
MIRQ-V in RISC-V

Beispiel Interruptverlauf



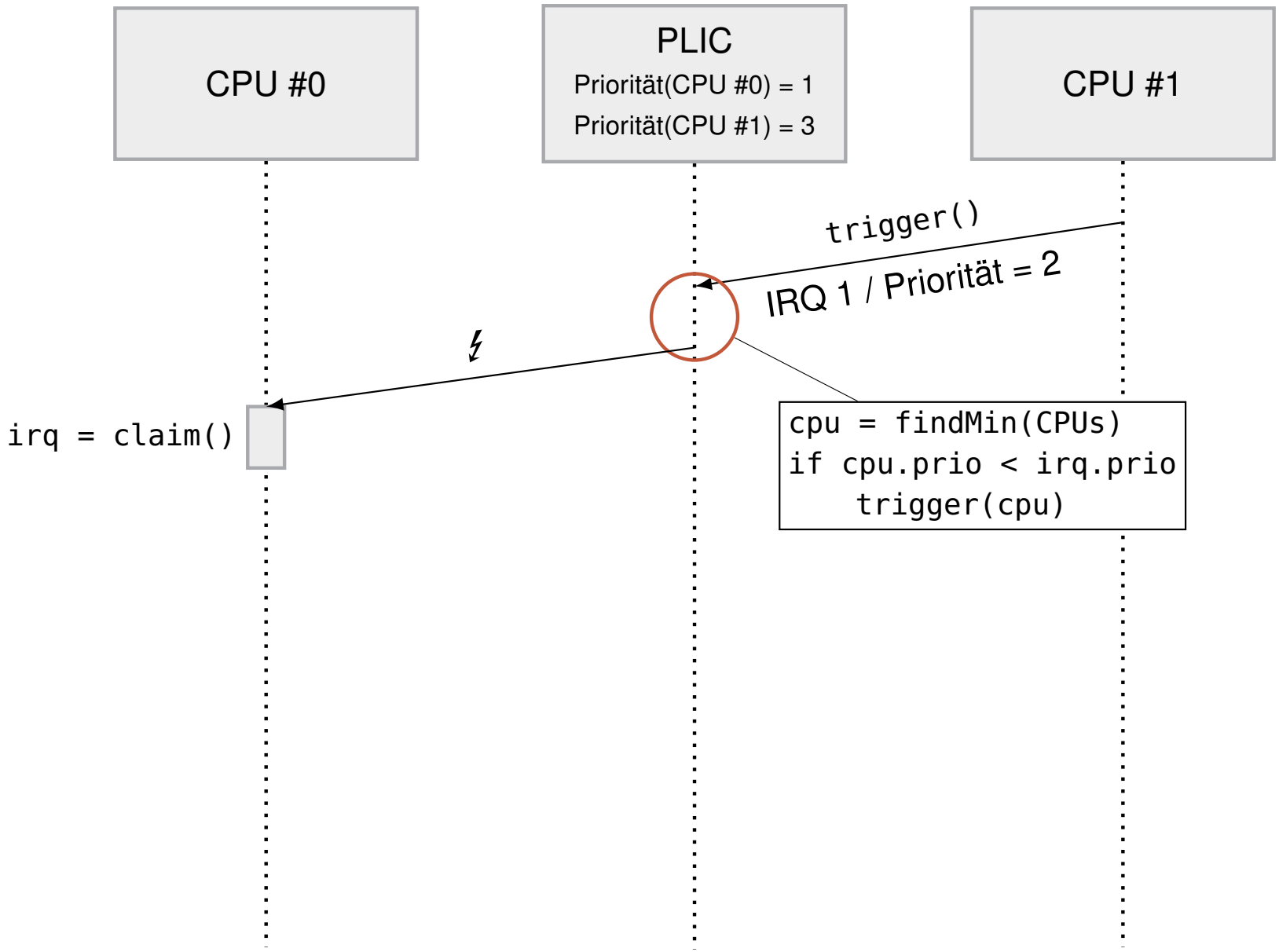
MIRQ-V in RISC-V

Beispiel Interruptverlauf



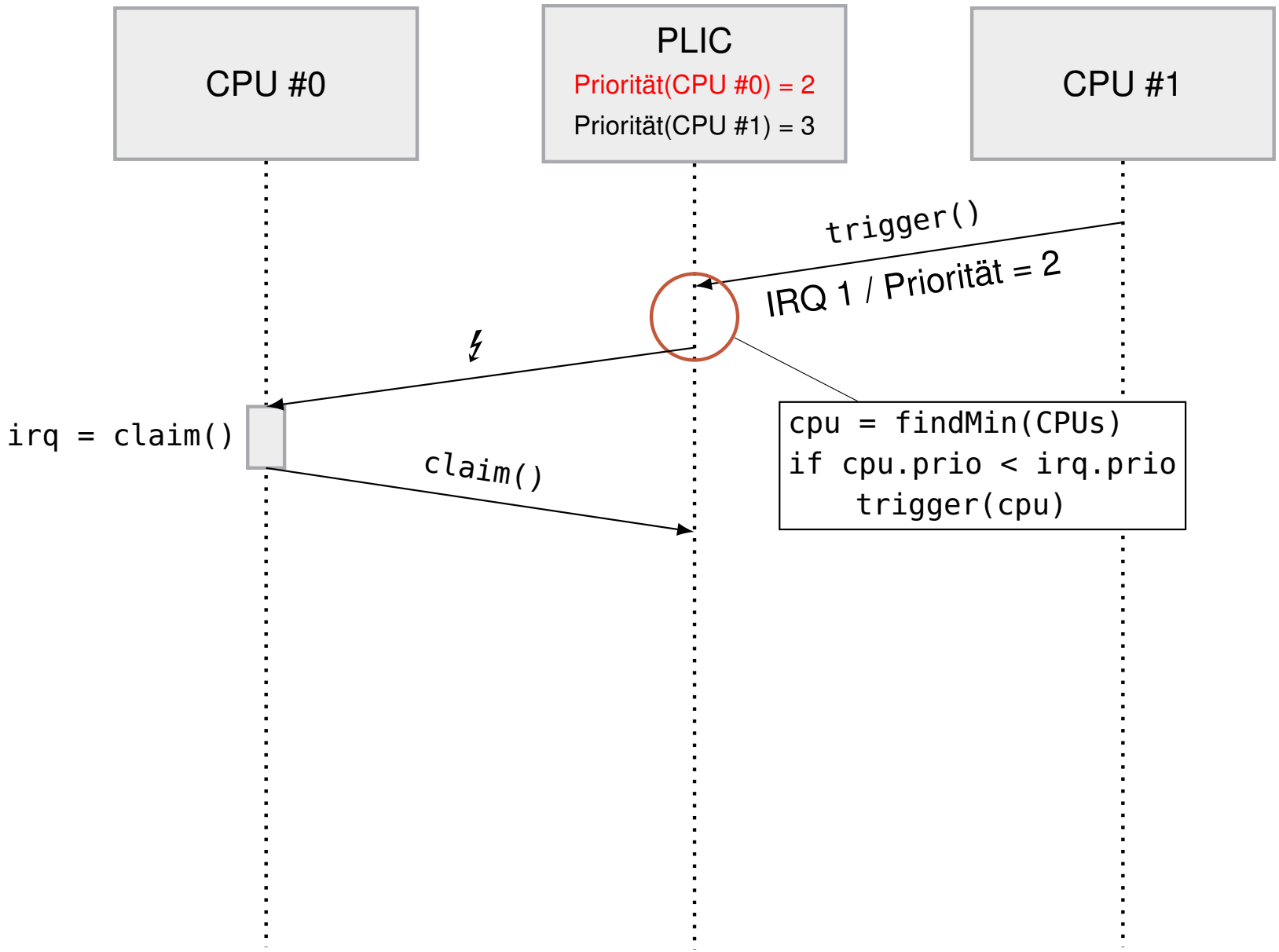
MIRQ-V in RISC-V

Beispiel Interruptverlauf



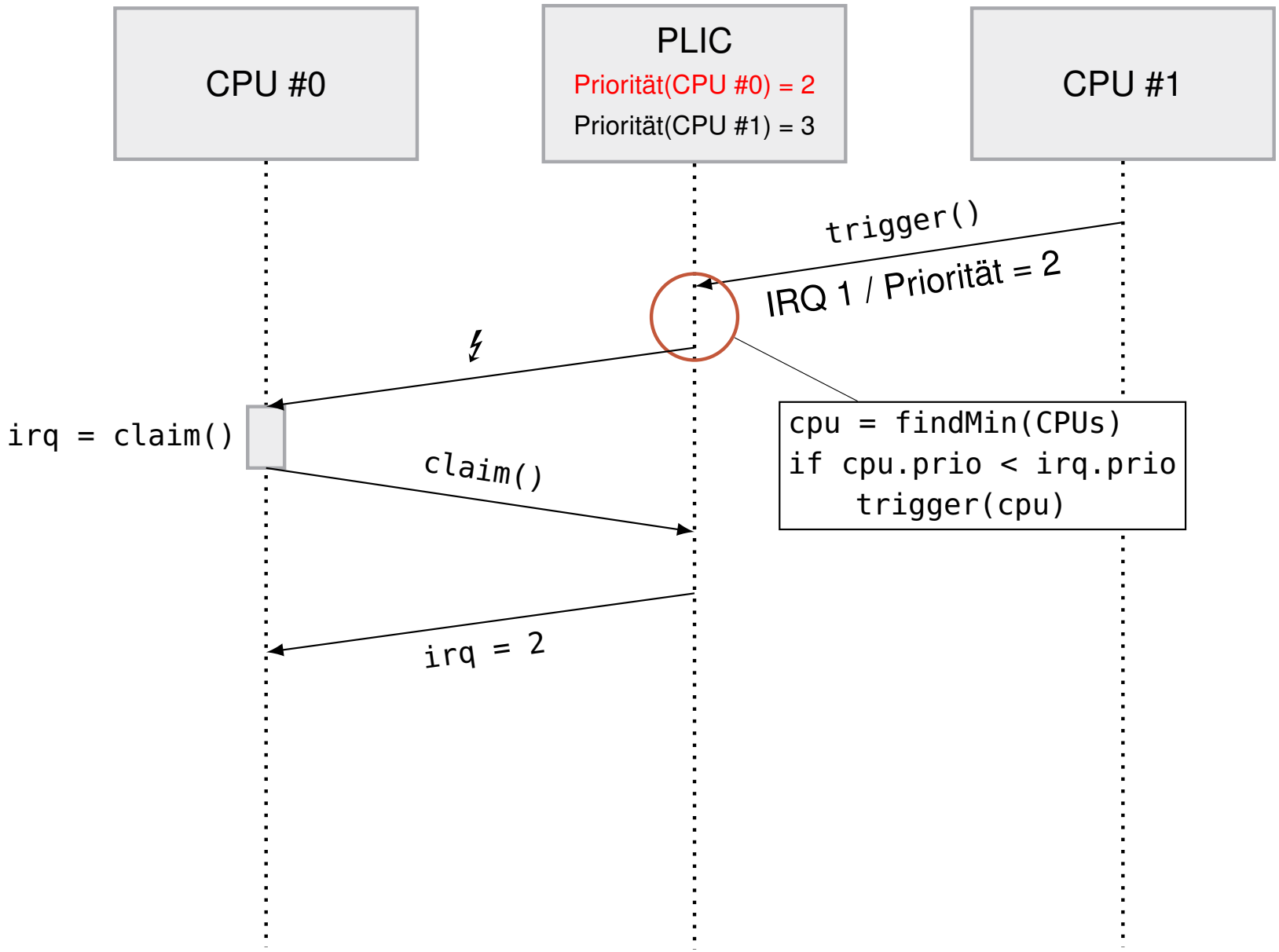
MIRQ-V in RISC-V

Beispiel Interruptverlauf



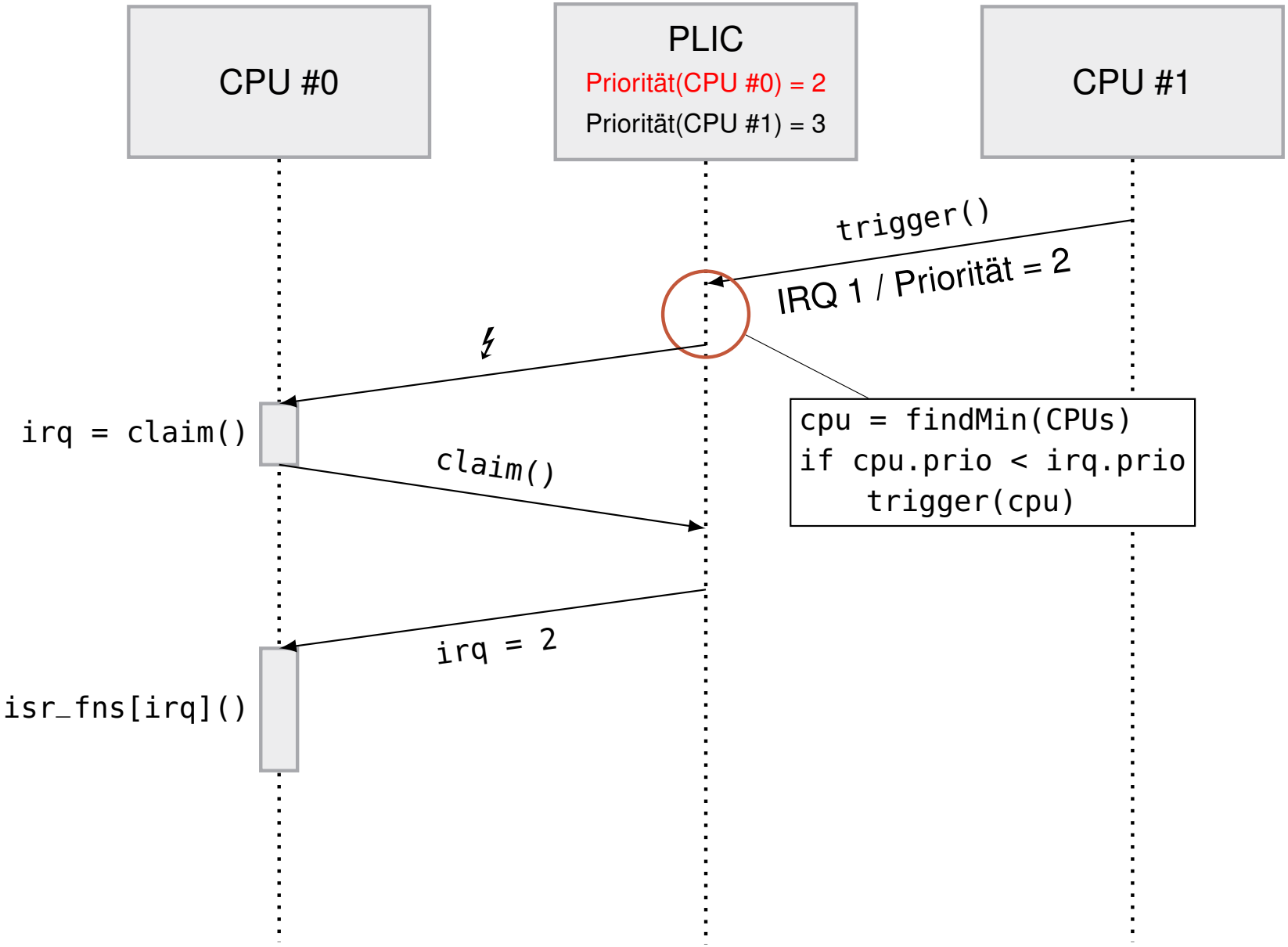
MIRQ-V in RISC-V

Beispiel Interruptverlauf



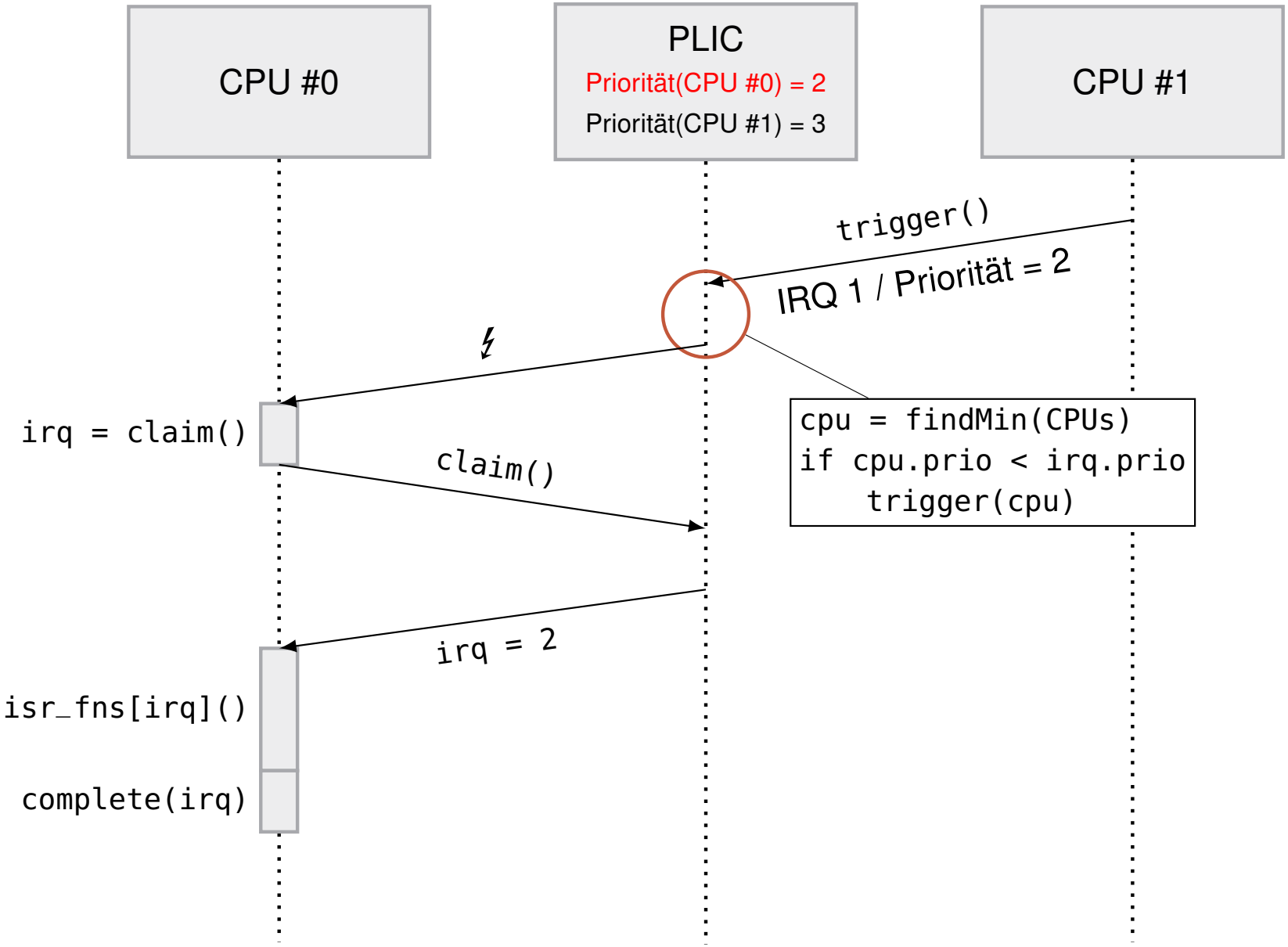
MIRQ-V in RISC-V

Beispiel Interruptverlauf



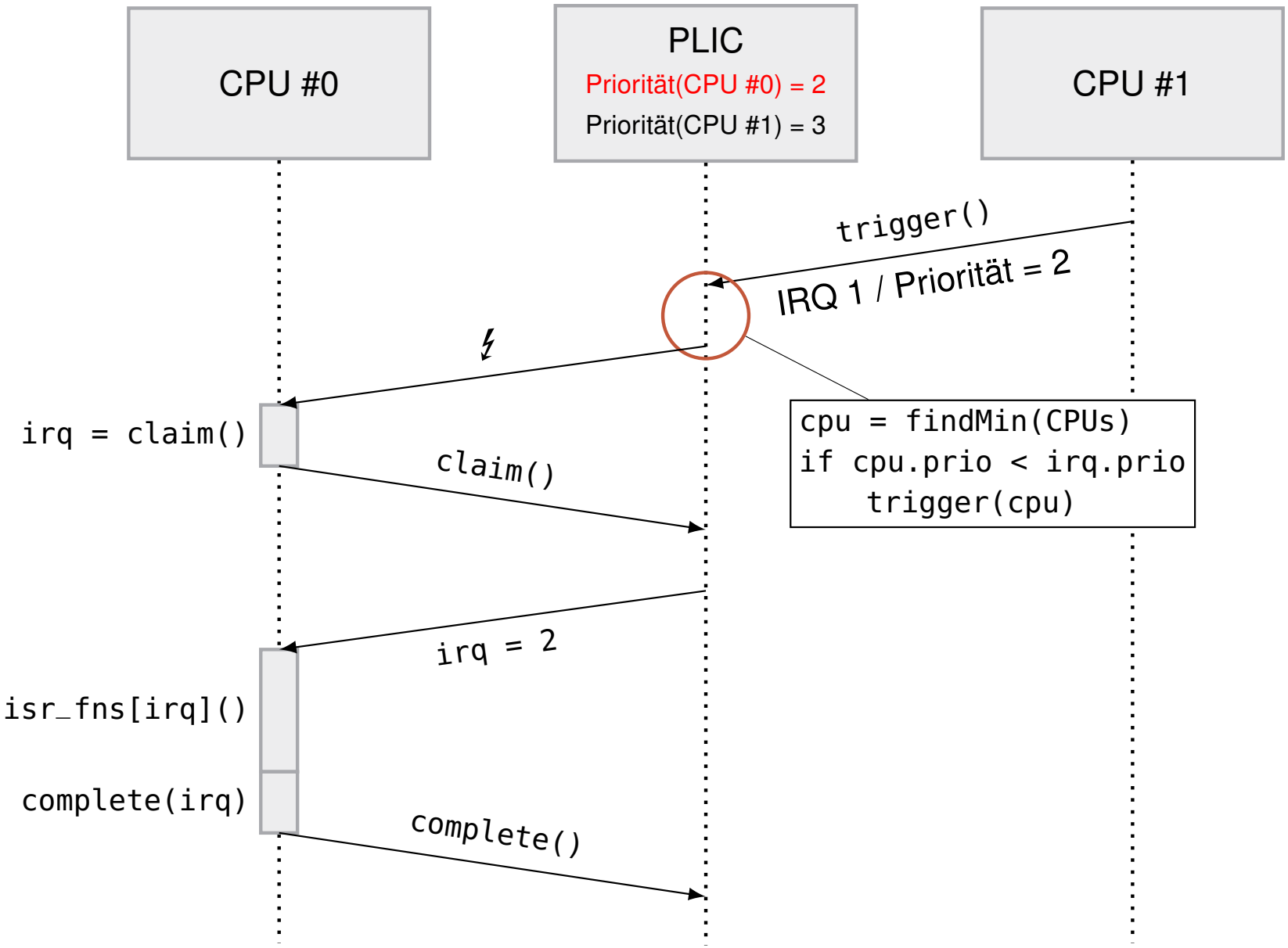
MIRQ-V in RISC-V

Beispiel Interruptverlauf



MIRQ-V in RISC-V

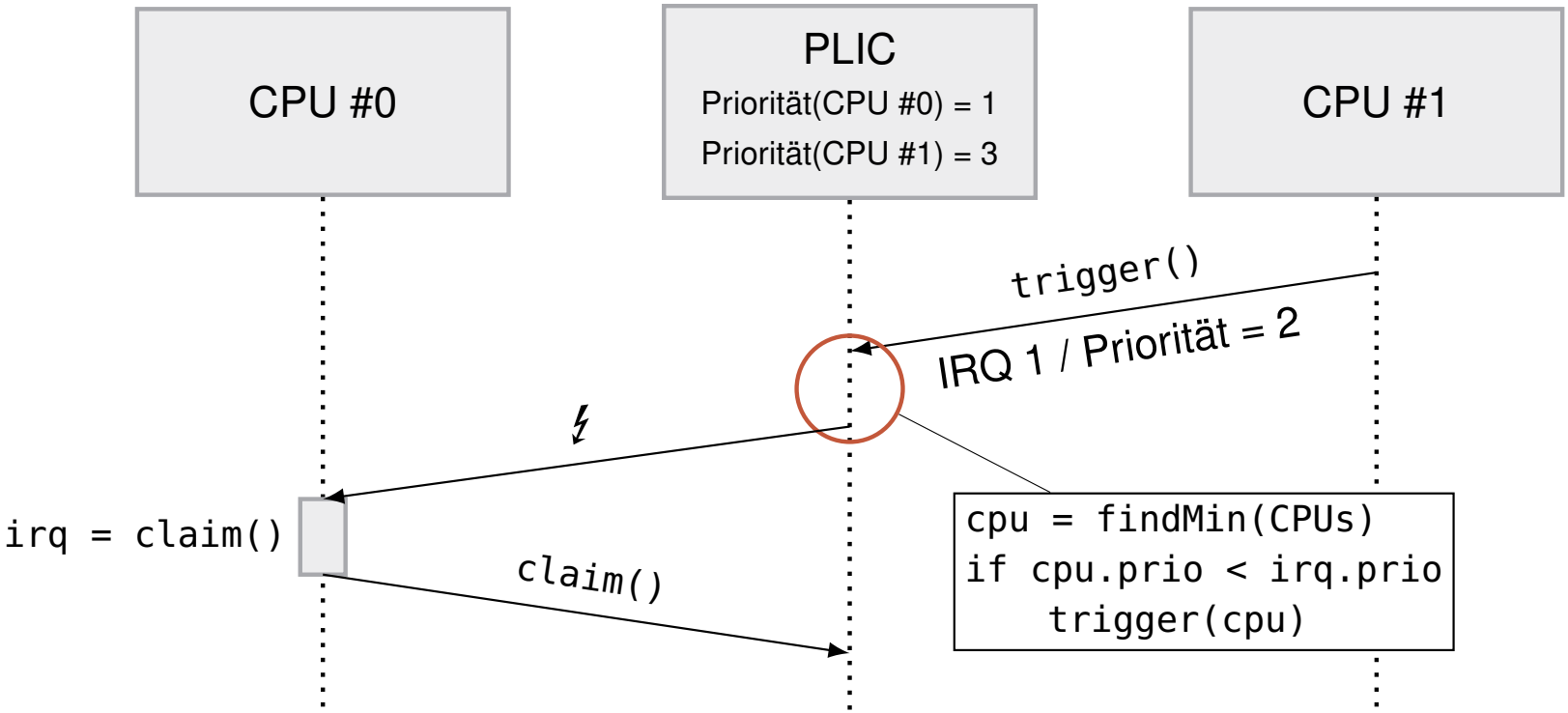
Beispiel Interruptverlauf



MIRQ-V in RISC-V

MIRQ-V in RISC-V

- Beachtung von Race-Condition bei Prioritätsänderung
  - Standardinterrupt hat zwei Zustände
    - Ausgelöst
    - In Bearbeitung
  - Neuer Zustand von Interrupt: Zugestellt
  - Bei Prioritätsänderung von Ziel-CPU automatisch neue Zustellung



MIRQ-V in RISC-V

- Beachtung von Race-Condition bei Prioritätsänderung
  - Standardinterrupt hat zwei Zustände
    - Ausgelöst
    - In Bearbeitung
  - Neuer Zustand von Interrupt: Zugestellt
  - Bei Prioritätsänderung von Ziel-CPU automatisch neue Zustellung
- Kodierung von Interruptoperationen in `claim-Register`
  - Erhaltung von Rückwartskompatibilität
  - 3 Operationen

`complete()`	
`00`	IRQ

`migrate()`	
`01`	IRQ

`trigger()`	
`10`	IRQ

Motivation

MIRQ-V

InterSloth

Evaluation

Zusammenfassung

InterSloth

InterSloth

- **Erinnerung:** Sloth $\Rightarrow$ Task $\equiv$ ISR

InterSloth

- **Erinnerung:** Sloth $\Rightarrow$ Task $\equiv$ ISR
- Implementierung von globalem Interruptscheduling trivial

InterSloth

- **Erinnerung:** Sloth $\Rightarrow$ Task $\equiv$ ISR
- Implementierung von globalem Interruptscheduling trivial
- Task kann jederzeit transparent verdrängt werden
  - Immer Kontextsicherung und/oder -wiederherstellung

InterSloth

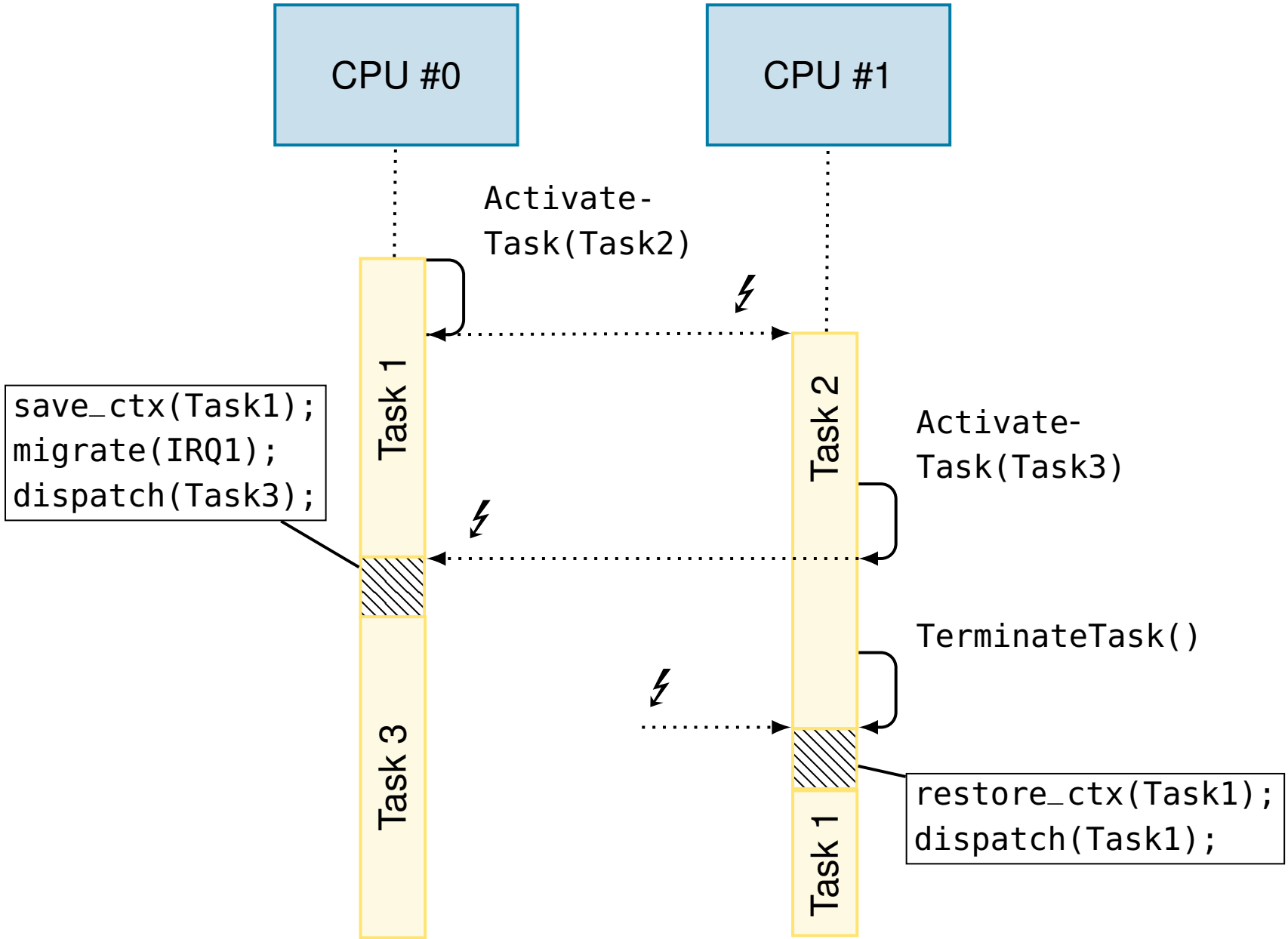
- **Erinnerung:** Sloth $\Rightarrow$ Task $\equiv$ ISR
- Implementierung von globalem Interruptscheduling trivial
- Task kann jederzeit transparent verdrängt werden
  - Immer Kontextsicherung und/oder -wiederherstellung
- **Problem:** Was passiert wenn bereits ein Interrupt auf CPU läuft?

InterSloth

- **Erinnerung:** Sloth $\Rightarrow$ Task $\equiv$ ISR
- Implementierung von globalem Interruptscheduling trivial
- Task kann jederzeit transparent verdrängt werden
  - Immer Kontextsicherung und/oder -wiederherstellung
- **Problem:** Was passiert wenn bereits ein Interrupt auf CPU läuft?
  - Migration auf andere CPU
  - Unterstützung durch Interruptcontroller nötig

InterSloth

Beispiel Unterbrechungsmigration



Motivation

MIRQ-V

InterSloth

Evaluation

Zusammenfassung

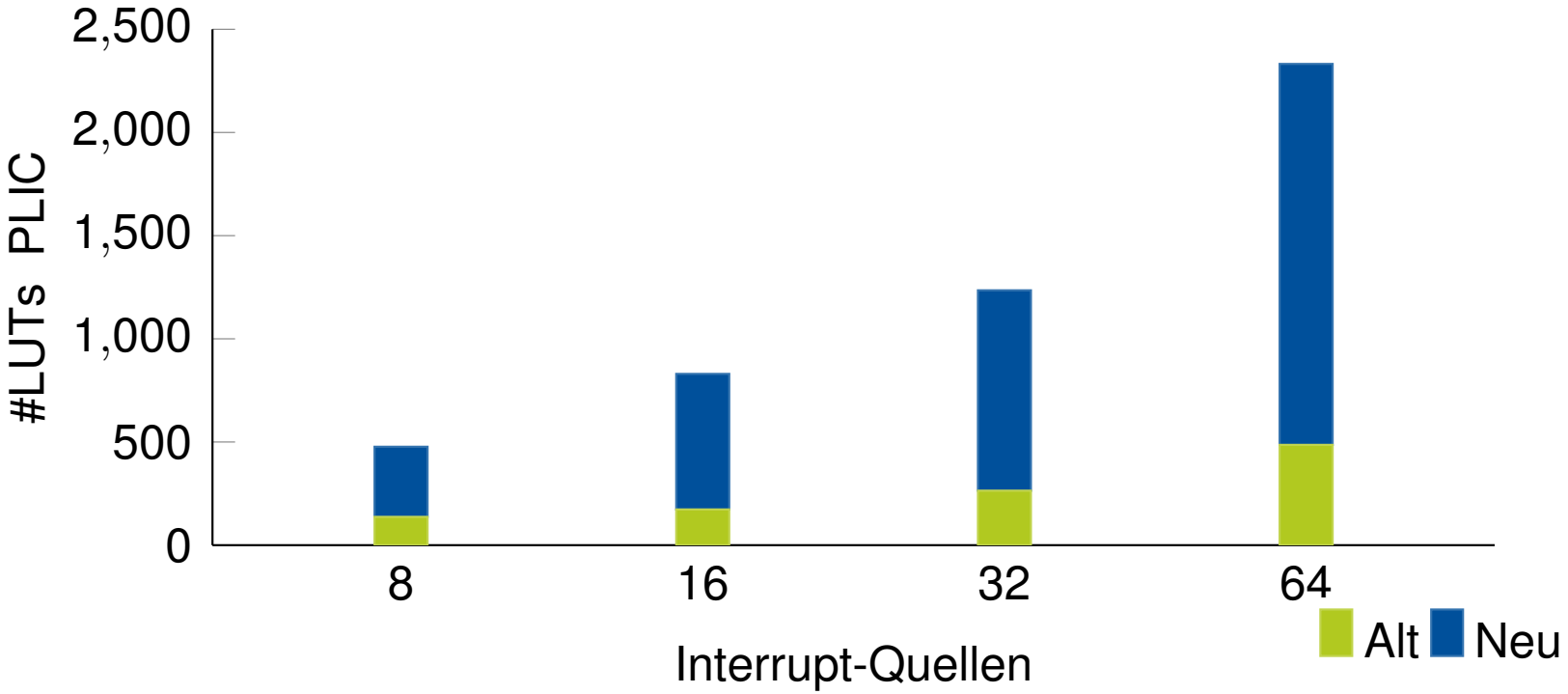
Evaluation

Evaluation

- Latenz von 1 Takt bis Interruptzustellung
  - MIRQ-V benötigt 4 Takte

Evaluation

- Latenz von 1 Takt bis Interruptzustellung
  - MIRQ-V benötigt 4 Takte
- Lineare Erhöhung von Basisbausteinen gegenüber Basis-PLIC
  - Abhängig von Anzahl an verfügbaren Interruptquellen
  - Gesamt `rocket-chip` verbraucht ca. 23.000 Basiszellen



Evaluation

Evaluation

- Messung in zyklen-akkuratem Simulator auf RT-Ebene (Verilator)
  - RISC-V stellt Performanceregister zur Verfügung (`mcycle`, ...)
  - Messung immer nur innerhalb eines Kerns möglich

Evaluation

- Messung in zyklen-akkuratem Simulator auf RT-Ebene (Verilator)
  - RISC-V stellt Performanceregister zur Verfügung (`mcycle`, ...)
  - Messung immer nur innerhalb eines Kerns möglich
- Messung von drei OSEK-Systemaufrufen

Evaluation

- Messung in zyklen-akkuratem Simulator auf RT-Ebene (Verilator)
  - RISC-V stellt Performanceregister zur Verfügung (`mcycle`, ...)
  - Messung immer nur innerhalb eines Kerns möglich
- Messung von drei OSEK-Systemaufrufen

Übergang	Takte	Instruktionen
`ActivateTask()`	318	130
`TerminateTask()` zu Leerlauf Task	93	60
`ChainTask()`	261	148

Zusammenfassung

Zusammenfassung

- Portierung von MIRQ-V auf RISC-V
  - Globale Prioritätstreue und Interruptmigration
  - Geringe Interruptlatenz von 1 Takt

Zusammenfassung

- Portierung von MIRQ-V auf RISC-V
  - Globale Prioritätstreue und Interruptmigration
  - Geringe Interruptlatenz von 1 Takt
- InterSloth
  - Globales FP-Scheduling ohne Synchronisierungsoverhead

Zusammenfassung

- Portierung von MIRQ-V auf RISC-V
  - Globale Prioritätstreue und Interruptmigration
  - Geringe Interruptlatenz von 1 Takt
- InterSloth
  - Globales FP-Scheduling ohne Synchronisierungsoverhead
- Zukünftige Forschung
  - Zusammenhang zwischen Hardwareaufwand und Interruptlatenz in MIRQ-V
  - Weitere Teile der OSEK Spezifikation: Ressourcen, Alarme, Events

Quellen

[Bew16] Christian Bewermeyer. “Priority-Obedient Multicore Interrupt Controller”. Bachelor Thesis. Friedrich-Alexander University Erlangen-Nuremberg, Nov. 2016.

[Bra11] Björn B. Brandenburg. “Scheduling and Locking in Multiprocessor Real-Time Operating Systems”. PhD thesis. The University of North Carolina at Chapel Hill, 2011. URL: http://www.cs.unc.edu/~bbb/diss/.

[Hof+09] Wanja Hofer et al. “Sloth: Threads as Interrupts”. In: *Proceedings of the 30th IEEE International Symposium on Real-Time Systems (RTSS ’09)* (Washington, D.C., USA, Dec. 1–4, 2009). IEEE Computer Society Press, Dec. 2009, pp. 204–213. ISBN: 978-0-7695-3875-4. DOI: 10.1109/RTSS.2009.18.

[Mül+14] Rainer Müller et al. “MultiSloth: An Efficient Multi-Core RTOS using Hardware-Based Scheduling”. In: *Proceedings of the 26th Euromicro Conference on Real-Time Systems (ECRTS ’14)* (Madrid, Spain). Washington, DC, USA: IEEE Computer Society Press, 2014, pp. 289–198. ISBN: 978-1-4799-5798-9. DOI: 10.1109/ECRTS.2014.30.