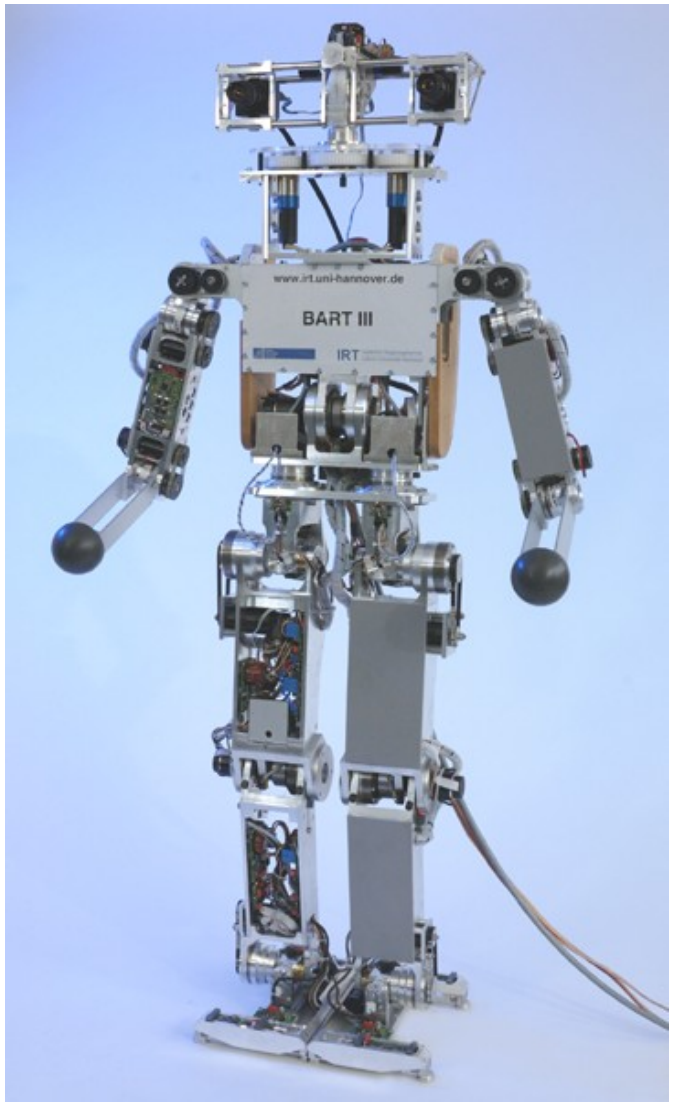
Universeller hybrider Systembeobachter für Echtzeitsysteme

Björn Pietsch
Institut für Regelungstechnik
Leibniz Universität Hannover



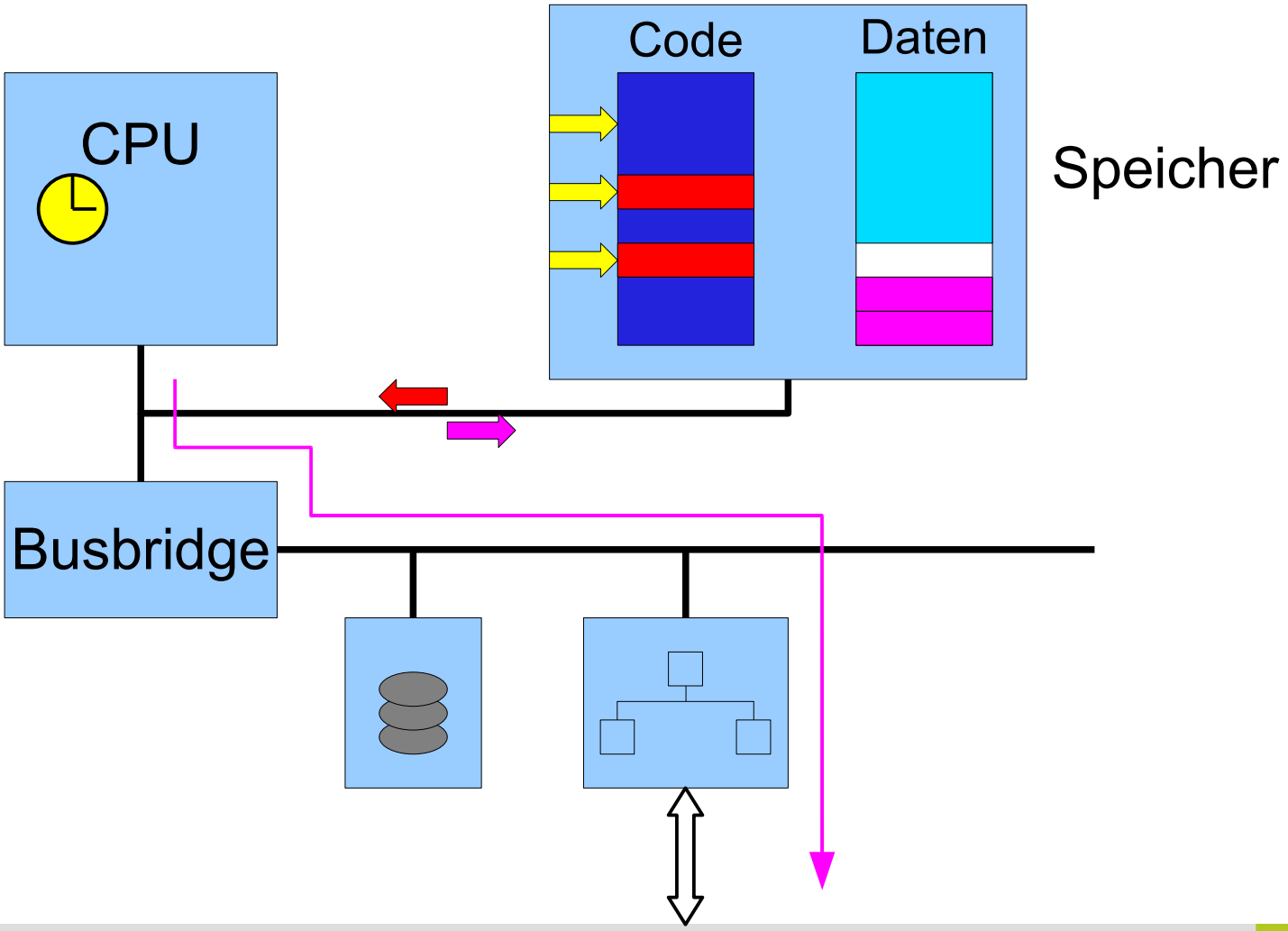
- eingebettete Systeme werden komplexer
- Vielzahl an Aufgaben
- Auslegung, Verifikation, Optimierung und Fehlersuche aufwendig
- Einsicht in innere Vorgänge notwendig

- Zeitverhalten von außen vorgegeben
- gleichzeitige Verfolgung interner und externer Abläufe nützlich
- herkömmliche Analysatoren und Debugger nur sehr bedingt geeignet

Gliederung

- Notwendigkeit der Aktionsverfolgung
- Grundtypen der Beobachter
- OSOBS: **O**perating **S**ystem **Obs**erver
- Effizienz und Anwendungen
- Ausblick

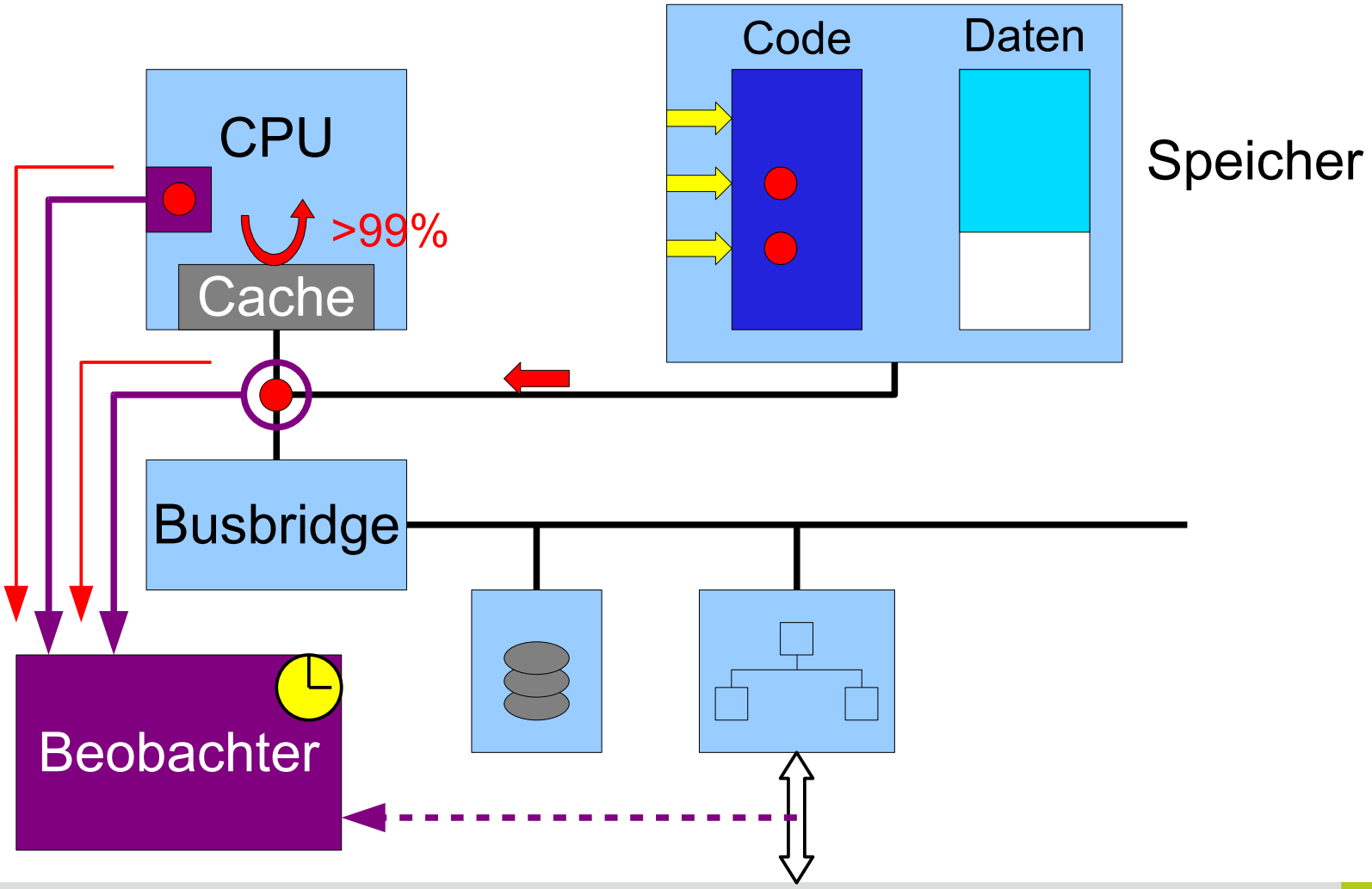
Softwarebeobachter



Softwarebeobachter

- systemeigene Tracer (z.B. ftrace, LTTng)
- keine zusätzliche Hardware
- leicht portierbar
- beanspruchen Systemressourcen
- Auflösung und/oder Aufzeichnungsdauer begrenzt
- erzeugen zusätzliche Ereignisse
- verändern Abläufe
- Beobachtung externer Vorgänge nicht möglich

Hardwarebeobachter

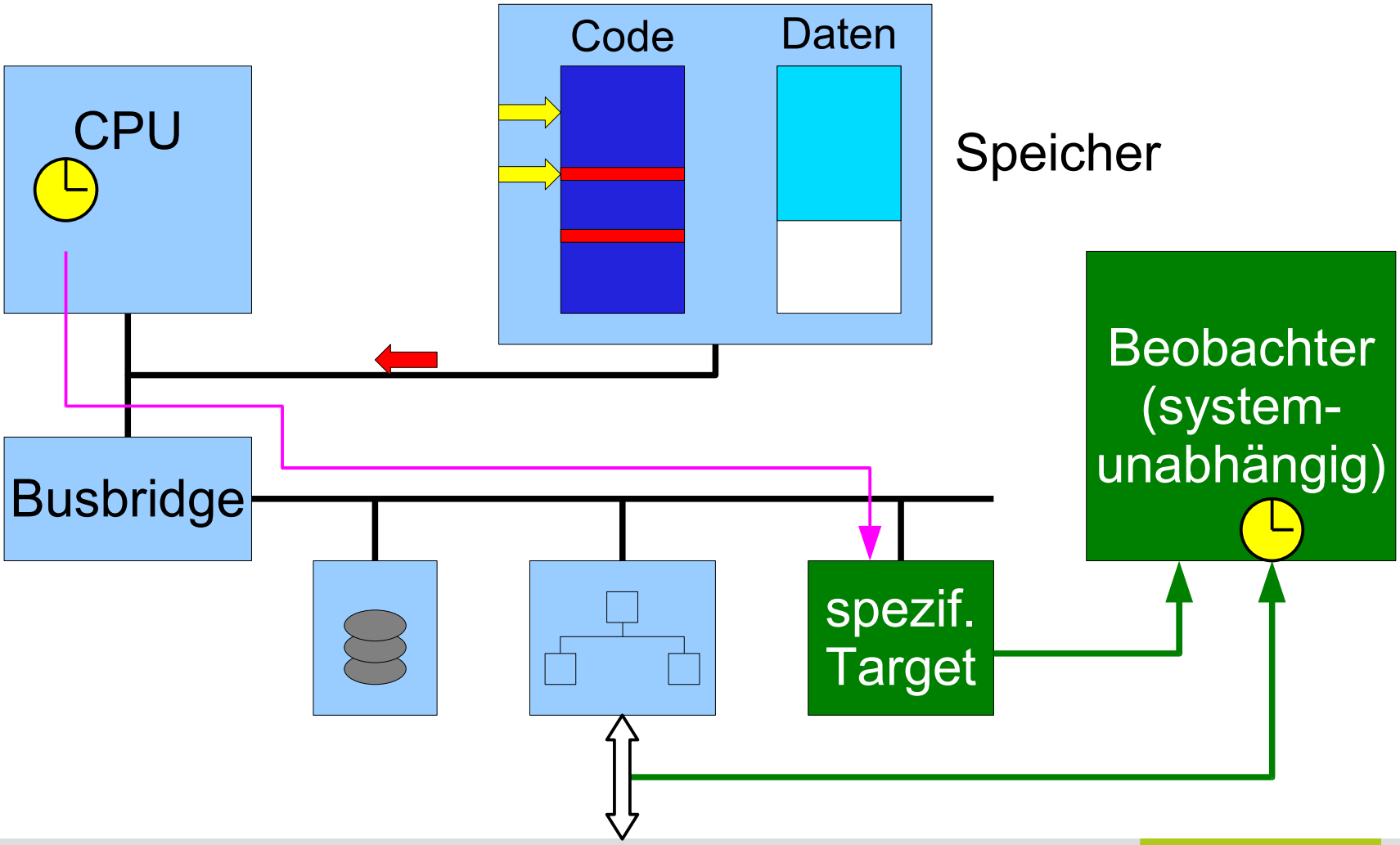


Hardwarebeobachter

- Busbeobachtung
  - rückwirkungsfrei bei zugänglichem Bus
  - System on Chip nutzt keinen externen Bus
  - Cache fängt Großteil der Zugriffe ab
  - komplexe Hardware notwendig

- Prozessormodule (z.B. Nexus5001, ARM Trace Port)
  - meist rückwirkungsfrei, keine Cache/SoC-Probleme
  - Daten unspezifisch: hohe Datenraten
  - komplexe Hardware notwendig

Hybrider Beobachter



Hybrider Beobachter

- Aufzeichnung mit externer Hardware
  - Datenumfang praktisch unbegrenzt
  - ständige Bereitschaft: einfacher Zugriff
  - einfache Überwachung peripherer Systeme
  - Hardware notwendig, aber standardisierbar
  - zwei Zeitbasen: Vereinheitlichung notwendig
- Aktive Ausgabe von Meldungen an wichtigen Stellen
  - Beschränkung auf bedeutende Ereignisse
  - standardisierte Datenkanäle (z.B. PCI, PCIe) nutzbar
  - Beeinflussung der Abläufe, aber gering

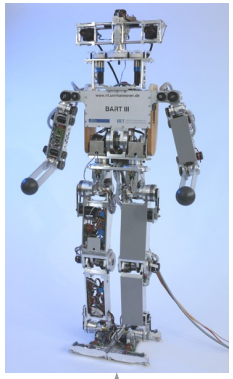
Anforderungen an einen hybriden Beobachter

- möglichst geringe Beeinflussung des Systems
- Unabhängigkeit von spezieller Softwarestruktur

- Modularität zur einfachen Anpassung an verschiedene Systeme
- Beobachtung von Netzwerken
- Erfassung und Anregung von Interrupts

- Widerstandsfähigkeit gegen Ereignisbursts
- Sicherung gegen Übertragungsfehler

OSOBS: Struktur am Beispiel eines Roboters



Daten

- Kontinuierliche Ereignisrate: 1,25 Mio./s
- Maximale Ereignisrate über PCI: 3,3 Mio./s
- Pufferkapazität: 250.000 Ereignisse
- Ereignisformat (4x4 Byte):
  - Zeitstempel
  - Quellkennung (Prozess/Quelle)
  - Ereigniscode
  - Statusmeldung
- 2 Interruptdetektoren/-generatoren

Beobachtungspunkte

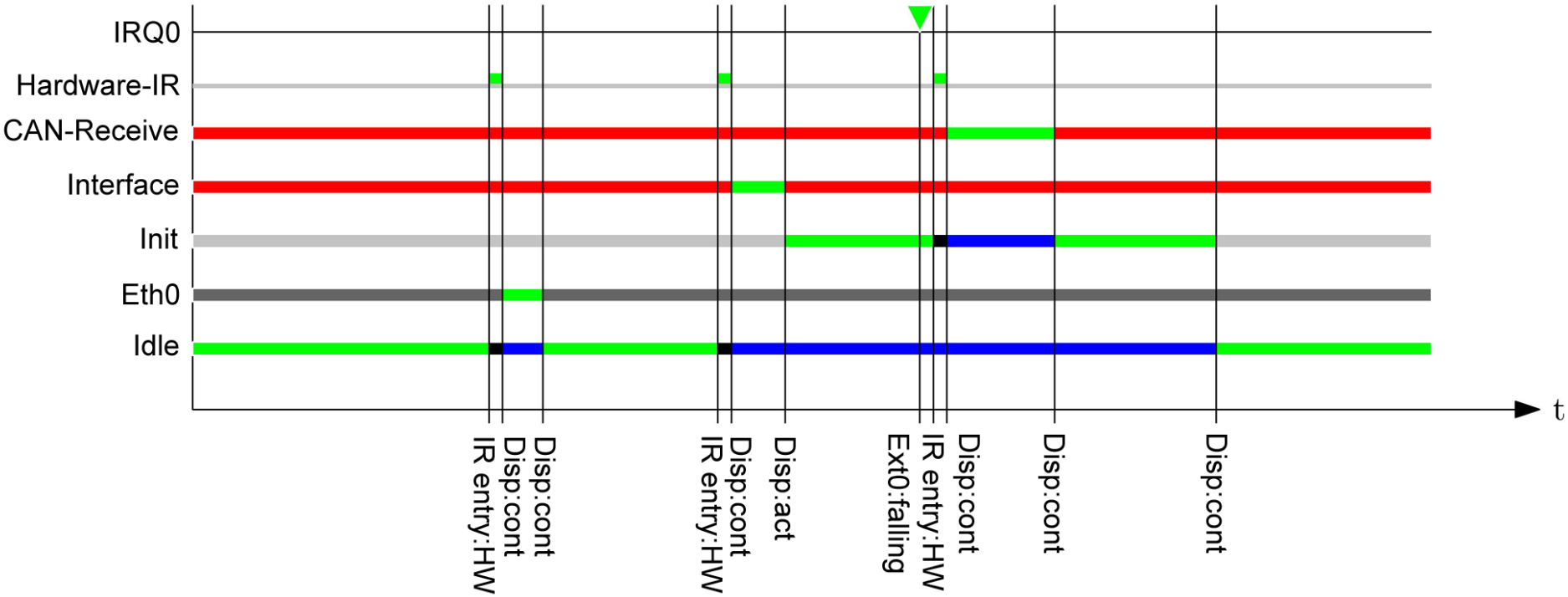
- Speicherfenster für Target, Adresse trägt auch Information
- Aufwand für Beobachtungspunkt (PowerPC):
  - 2 Speicherzugriffe
  - RTOS-UH: 5 Maschinenbefehle
  - Linux: 13 Maschinenbefehle, Optimierungspotential
- spezielle Ereigniscodes ermöglichen Datentransport:
  - Speicherbelegung
  - benutzerdefinierte Werte aus Anwendung

```
<event idx="1704">
        <time>374AF8C</time>
        <OID>14E90</OID>
        <EID>40002010</EID>
        <stat>0</stat>
        <mem>200</mem>
</event>
```

Analyse

- Vorverarbeitung
  - Konversion in XML
  - Wiederherstellung unterbrochener Beobachtungspunkte
  - Vereinheitlichung der Zeitmessung
  - Umwandlung der Datenereignisse
  - Zustandsrekonstruktion für Prozesse
- statistisch
  - gerufene Funktionen
  - Ausführungs- und Latenzzeiten
- grafisch

Grafische Analyse

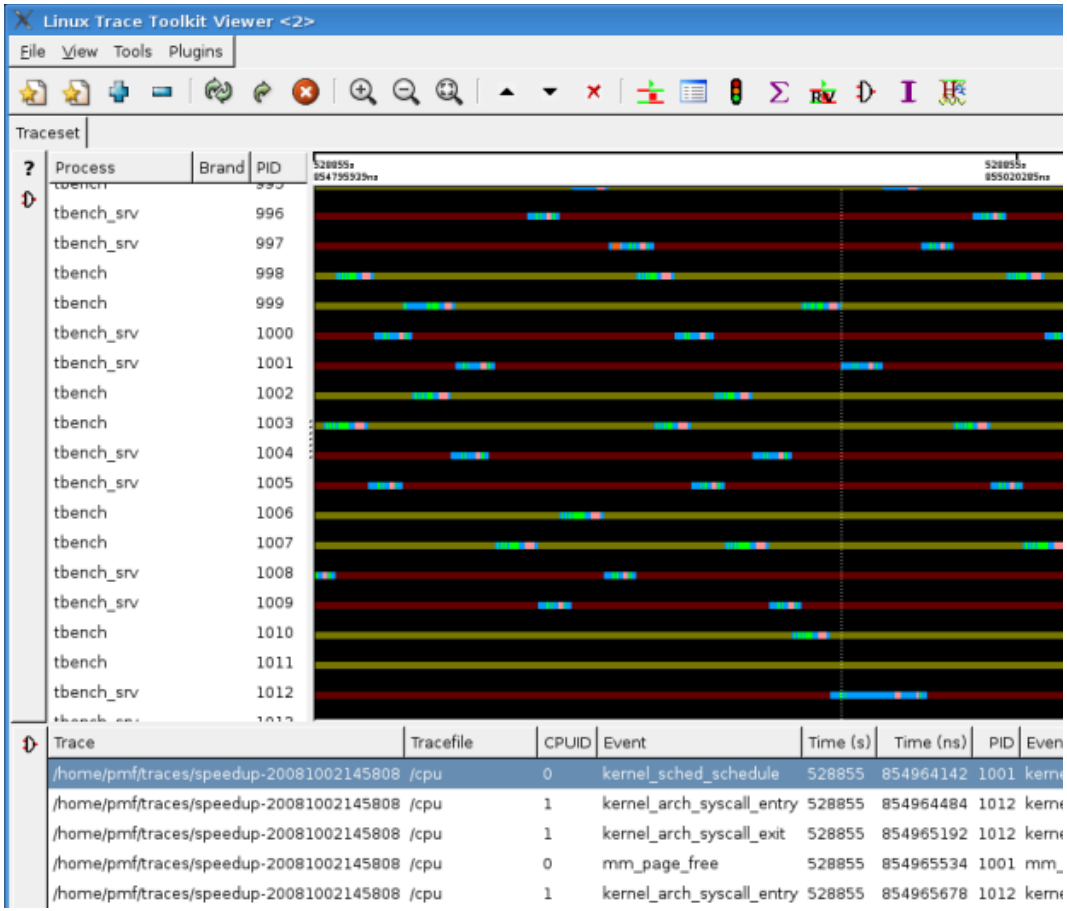


Anwendungen

- Treibereffizienz CAN
- Funktionsklärung des Gatekeepers in Linux/Xenomai
- Referenzerfassung des Ressourcenverbrauchs
- Versagensmechanismus bei Interruptlast

Ausblick

- Bedienkomfort
- Analysen fortführen
- LTTV-Integration



Quelle: www.lttng.org

Vielen Dank für die Aufmerksamkeit

Fragen ?

Björn Pietsch
Institut für Regelungstechnik
Leibniz Universität Hannover
www.irt.uni-hannover.de