www.lachmann-rink.de

LACHMANN & RINK | PARTNER FOR SMART INDUSTRIAL SOLUTIONS

Multiprozess-Echtzeitsysteme unter Linux

Der Weg zu minimaler Systemlatenz in komplexen Steuerungssystemen

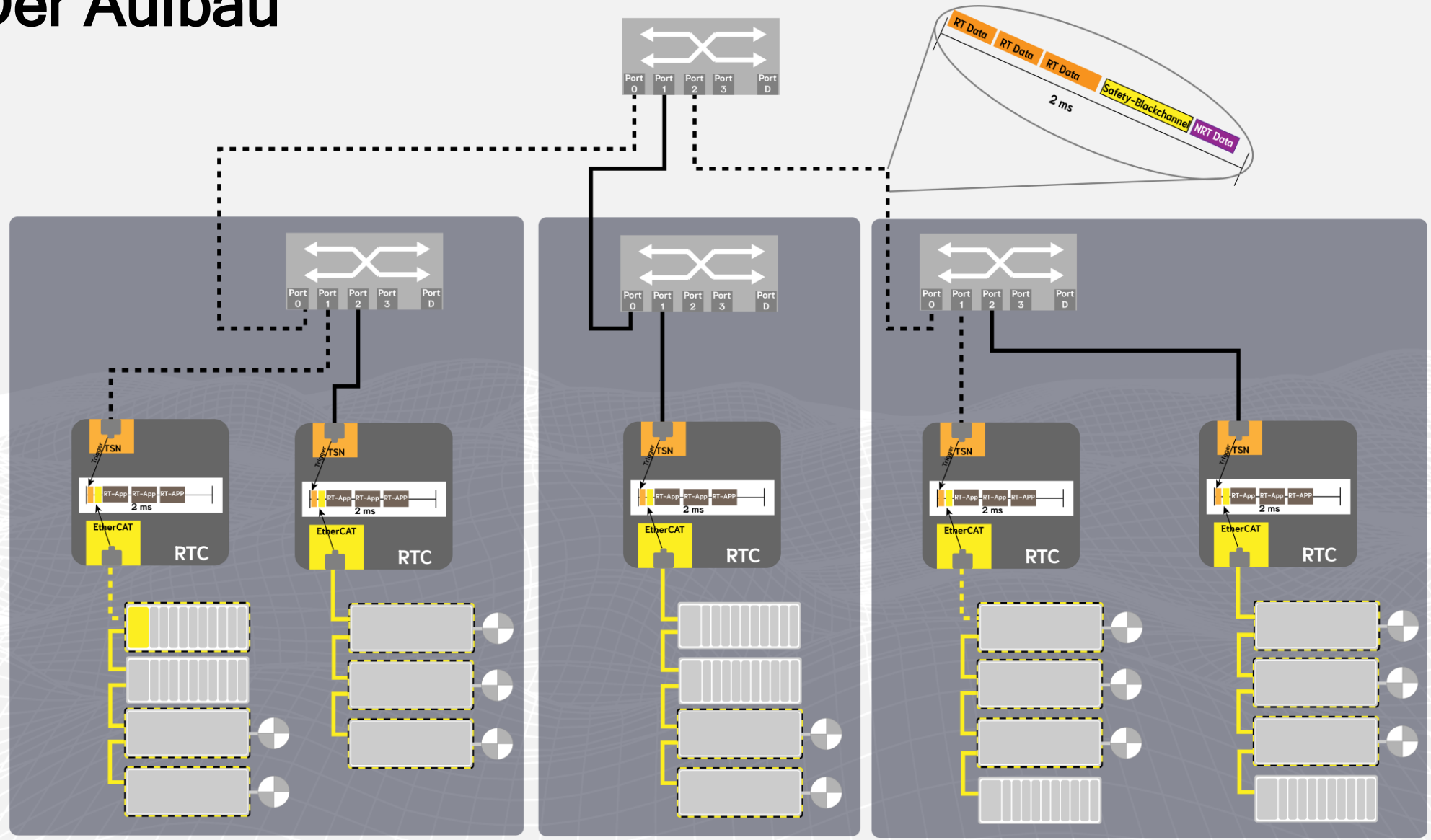
Agenda

- Projektbeispiel
- Bausteine von Echtzeit-Linux
- System Setup
- Applikation (en)
- Entwicklungs-Infrastruktur

Projektbeispiel

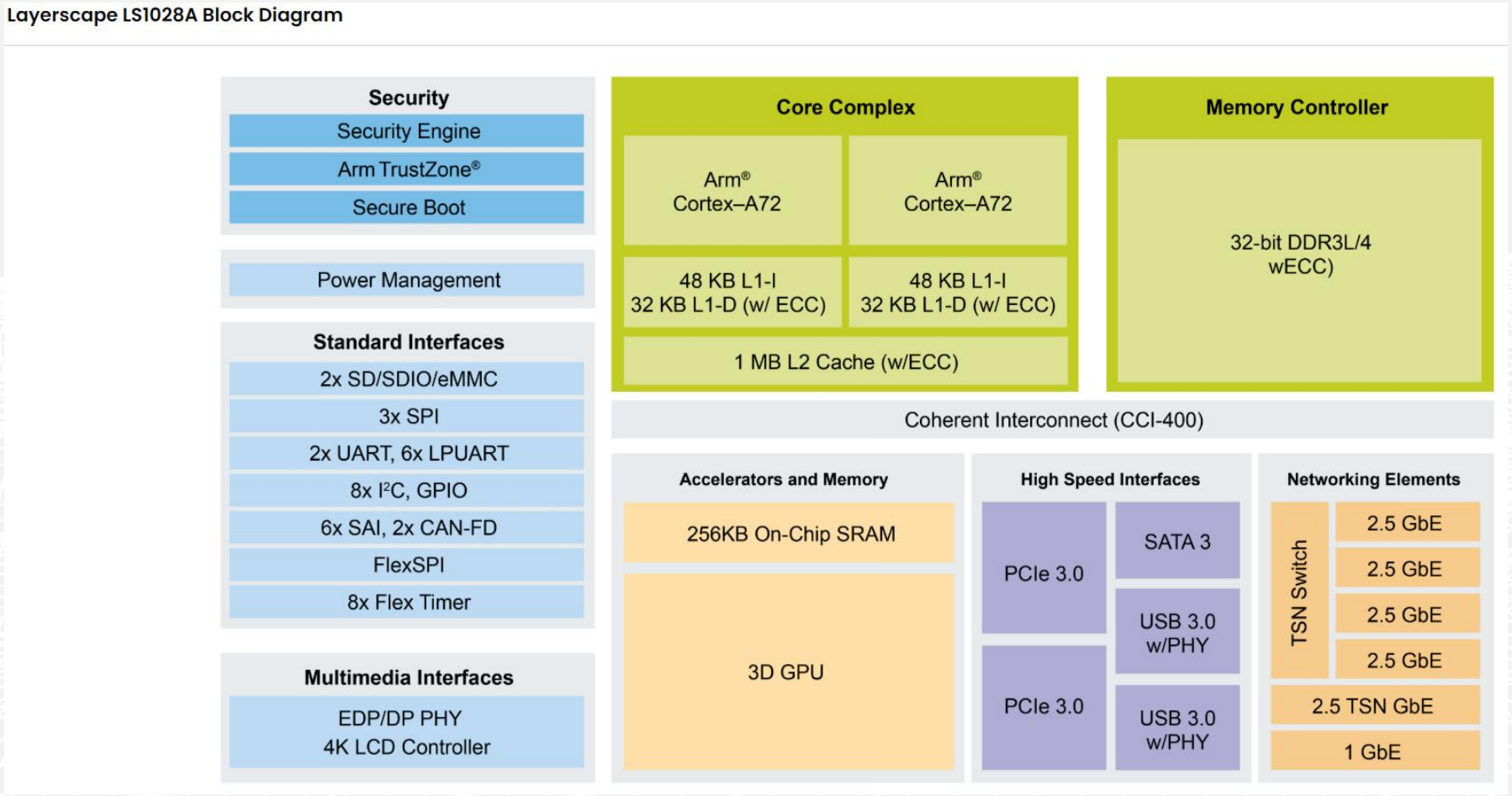
Verteiltes System - Zyklussynchron

Der Aufbau


Port 0
Port 1
Port 2
Port 3
Port D
RT Data
RT Data
RT Data
Safety-Blackchannel
NRT Data
2 ms
TSN
Trigger
RT-App
RT-App
RT-APP
2 ms
EtherCAT
RTC

Der Aufbau

Der Aufbau


Layerscape LS1028A Block Diagram
Security
Security Engine
Arm TrustZone®
Secure Boot
Power Management
Standard Interfaces
2x SD/SDIO/eMMC
3x SPI
2x UART, 6x LPUART
8x I²C, GPIO
6x SAI, 2x CAN-FD
FlexSPI
8x Flex Timer
Multimedia Interfaces
EDP/DP PHY
4K LCD Controller
Core Complex
Arm® Cortex–A72
Arm® Cortex–A72
48 KB L1-I 32 KB L1-D (w/ ECC)
48 KB L1-I 32 KB L1-D (w/ ECC)
1 MB L2 Cache (w/ECC)
Memory Controller
32-bit DDR3L/4 wECC)
Coherent Interconnect (CCI-400)
Accelerators and Memory
256KB On-Chip SRAM
3D GPU
High Speed Interfaces
PCIe 3.0
PCIe 3.0
SATA 3
USB 3.0 w/PHY
USB 3.0 w/PHY
Networking Elements
TSN Switch
2.5 GbE
2.5 GbE
2.5 GbE
2.5 GbE
2.5 TSN GbE
1 GbE

Zum Projekt

- Verteile Anlage
- Controller to Controller Kommunikation via TSN (Time Sensitive Networking)
- Controller arbeiten Anlagenweit zyklussynchron
- Zykluszeit 2ms
- N Echtzeit-Applikationen pro Steuerung (Hier nur eine)
- M Nicht-Echtzeit-Applikationen pro Steuerung

Weitere Herausforderungen

- M Applikationen → M Teams
  - RT-Setup und Test des Kernels und der Hardware
  - Jede Applikation hat ein eigenes Entwicklungsteam
  - „Dritte“ Entwickeln potenziell eigene Applikationen
- Konsistente Updates von Hardware-Images und Dev-Umgebungen
- Reproduzierbarkeit von Auslieferungen muss gegeben sein

“Die Kunst ist, das Ganze im Auge zu behalten und dennoch über Details zu streiten“

- Unbekannt -

Agenda

- Projektbeispiel
- Bausteine von Echtzeit-Linux
- System Setup
- Applikation (en)
- Entwicklungs-Infrastruktur

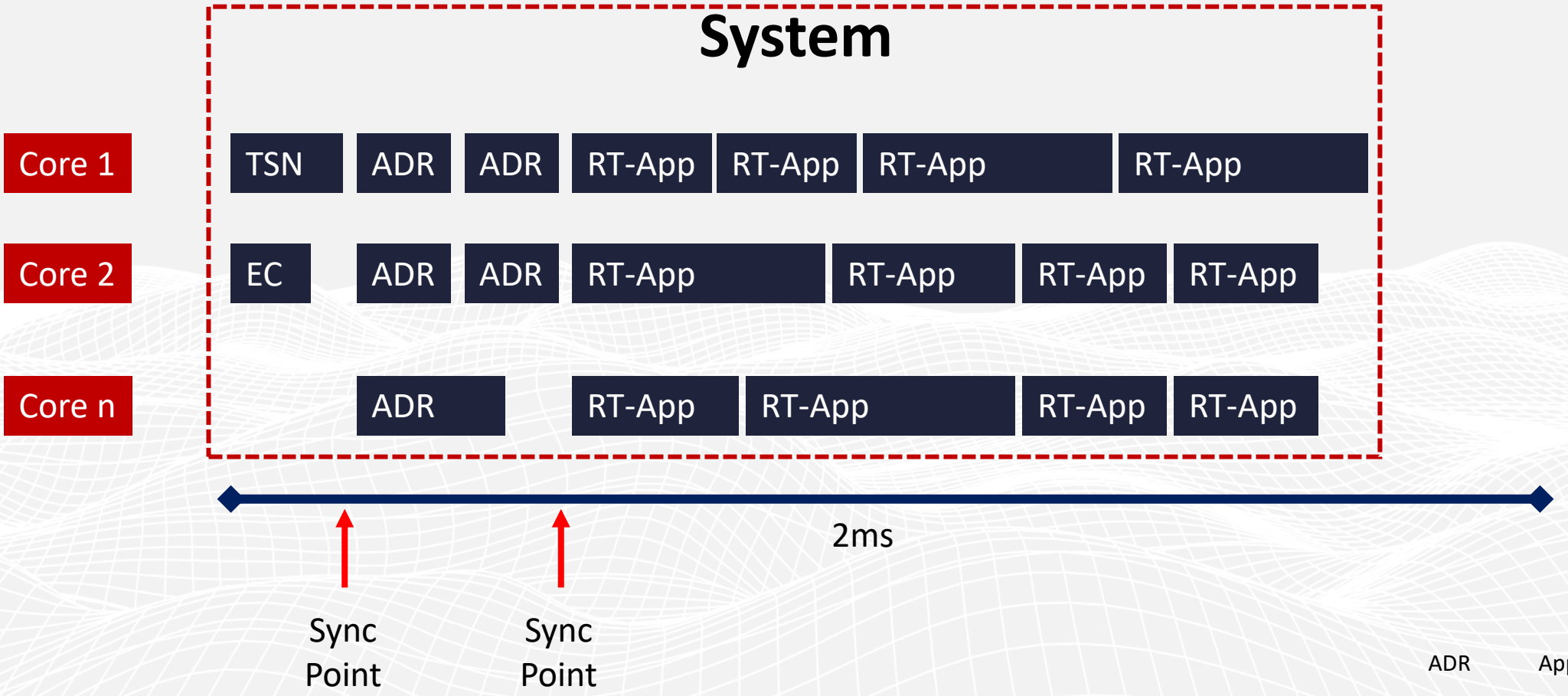
Recap

Echtzeit?

"A system is said to be real-time if the total correctness of an operation depends not only upon its logical correctness, but also upon the time in which it is performed"

- Shin, K.G.: Ramanathan, P. (Jan 1994) -

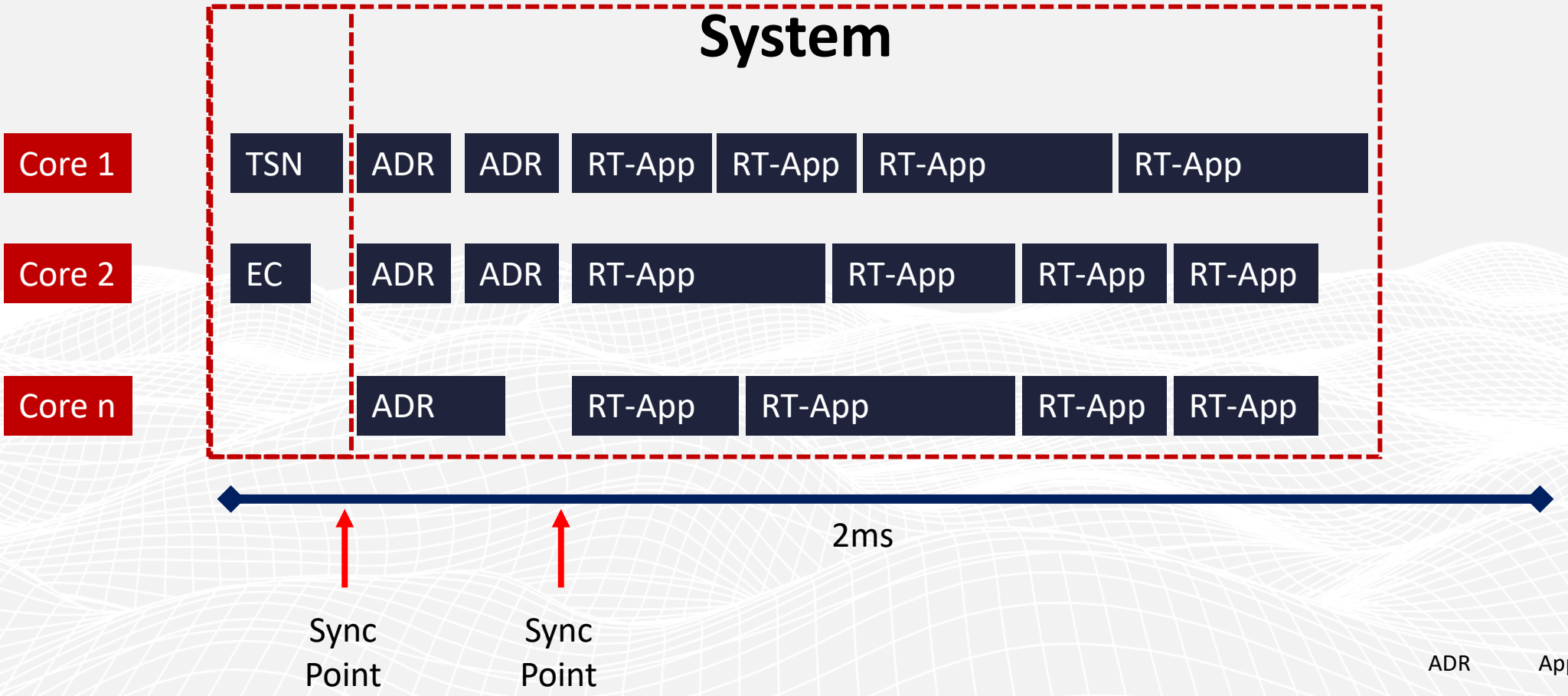
Echtzeit in diesem Kontext



ADR Appl. Driver

RT-APP App oder Thread

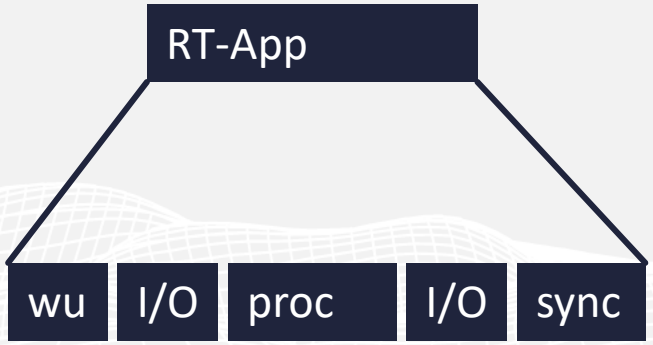
Echtzeit in diesem Kontext



ADR Appl. Driver

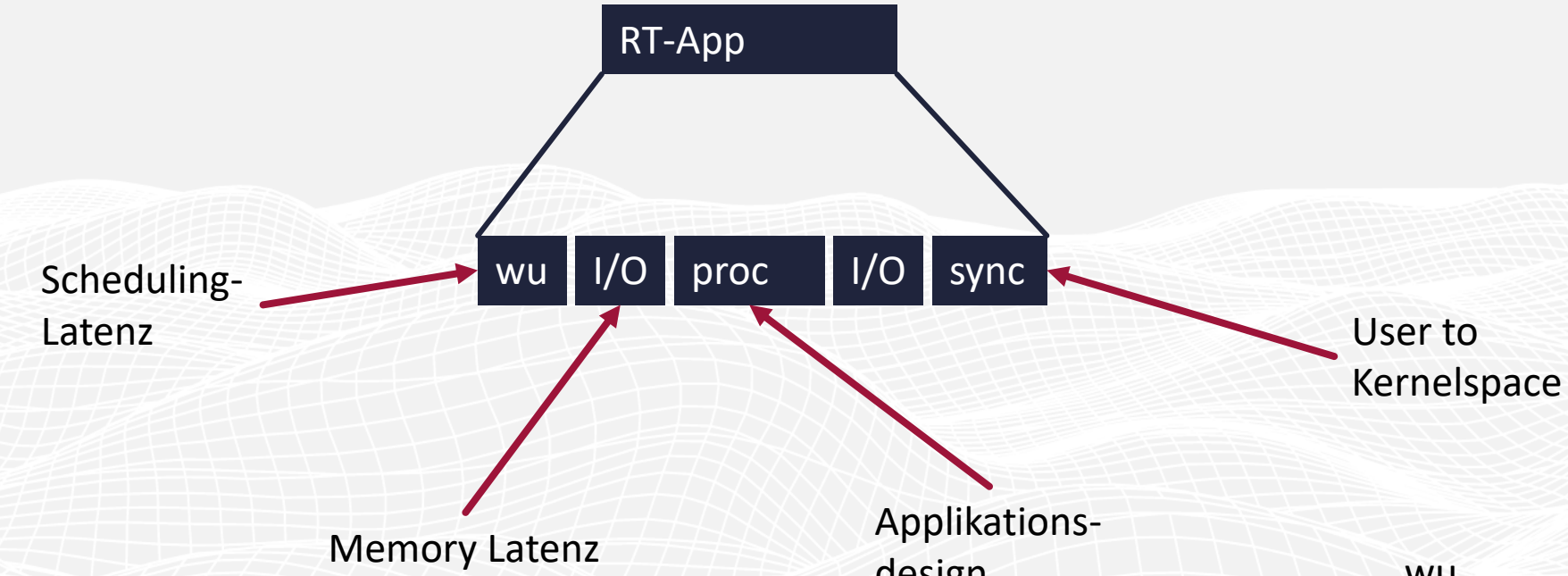
RT-APP App oder Thread

Echtzeit in diesem Kontext



wu → wakeup
proc → processing
sync → synchonisation

Echtzeit in diesem Kontext



wu → wakeup
proc → processing
sync → synchonisation

Ebenen

- Latenzen durch Hardware und Betriebssystem
- Design einer einzelnen Applikation
- “Choreografie“ von n Applikationen

Maßnahmen

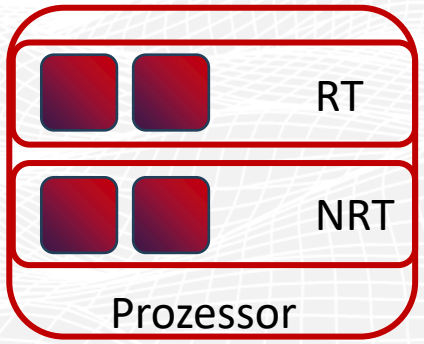
- Hardwareauswahl
- Build-Zeit → Kernel Konfiguration
- Boot-Zeit → Bootargs
- Laufzeit → cgroup, Prio, Memory
- Applikation → …

Agenda

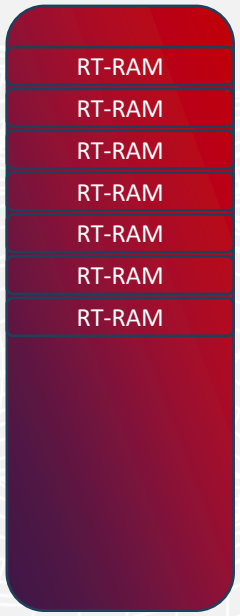
- Projektbeispiel
- Bausteine von Echtzeit-Linux
- System Setup
- Applikation (en)
- Entwicklungs-Infrastruktur

Die Steuerung - Linux-Konfiguration

CPU



Speicher



Interrupts

Power

Linux Setup

- Build-Zeit → Kernel Konfiguration
  - CONFIG_PREEMPT_RT_FULL Y
  - CONFIG_HIGH_RES_TIMERS Y
  - CONFIG_IRQ_FORCED_THREADING Y
  - CONFIG_CPU_FREQ_DEFAULT_GOV_PERFORMANCE Y
  - CONFIG_SUSPEND N

Linux Setup – Real-Life-Abwägungen

- Kann ich das Debian-Paket linux-image-rt-<arch> nutzen?
  - Weniger Kernel-Config und weniger Maintenance
  - Nicht für alle Boards bzw. meinen speziellen Use-Case möglich
- Muss ich den Kernel selbst erstellen?
  - Einbindung eigener Treiber
  - Freiheit den Kernel maximal auszudünnen
  - Maintenance-Aufwand höher (Updates u.ä.)
- Wird weniger relevant, aber:
  - So nah wie möglich an Mainline bleiben!!!

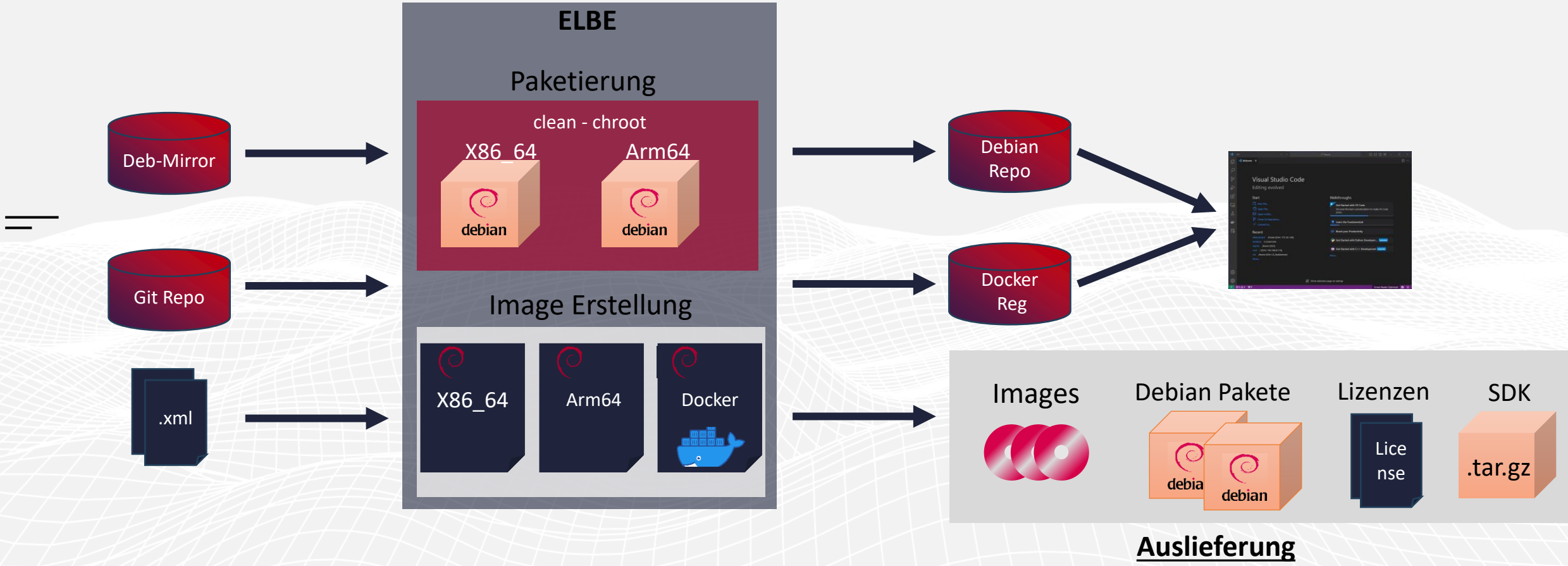
Linux Setup – Device-Tree-Anpassungen

- Custom Board → Custom Device-Tree
- Device-Tree Anpassung im Kernel vs. eigenes Paket
  - Anpassungen leichter bei eigenem Paket
  - Pflegeaufwand geringer bei Upstream-Updates
- Falls wenig Speicher auf System: Treiber, die im DT nicht verwendet werden aus Kernel werfen.

Einschub - Agenda

- Projektbeispiel
- Bausteine von Echtzeit-Linux
- System Setup
- Applikation (en)
- Entwicklungs-Infrastruktur

Build-Umgebung



https://elbe-rfs.org

Elbe – Systembeschreibung via XML

- Siehe elbe_files/01_first_image_rt.xml

Elbe – Arbeitsmodi

- Via elbe initvm (High-Level)
- Development-Mode via elbe control (Feingranular und inkrementell)

Elbe – Arbeitsmodus Development

```
export PRJ=$(elbe control create project)
echo $PRJ
elbe preprocess path/to/xml.xml
elbe control set_xml $PRJ preprocess.xml
elbe control build $PRJ && elbe control wait_busy $PRJ
elbe control get_files $PRJ
elbe control get_file $PRJ sdcard.img.tar.gz -output=./
```

Elbe - Development Änderungen vornehmen

```
# Änderungen an der xml vornehmen
elbe preprocess path/to/xml.xml
elbe control set_xml $PRJ preprocess.xml
elbe control build $PRJ && elbe control wait_busy $PRJ
elbe control get_files $PRJ
elbe control get_file $PRJ sdcard.img.tar.gz -output=./
```

Elbe – Development Änderungen vornehmen

- Änderungen, die nicht inkrementell übernommen werden
  - SD-Karten Image Größe (Neues Projekt notwendig)

Linux Setup

- Boot-Zeit → Bootargs
  - isolcpus=2-3
  - arm-smmu.disable_bypass=0
  - cpufreq.default_governor=performance
  - processor.max_cstate=0
  - irqaffinity=0,1
- Siehe elbe_files/03_minbase_rt.xml

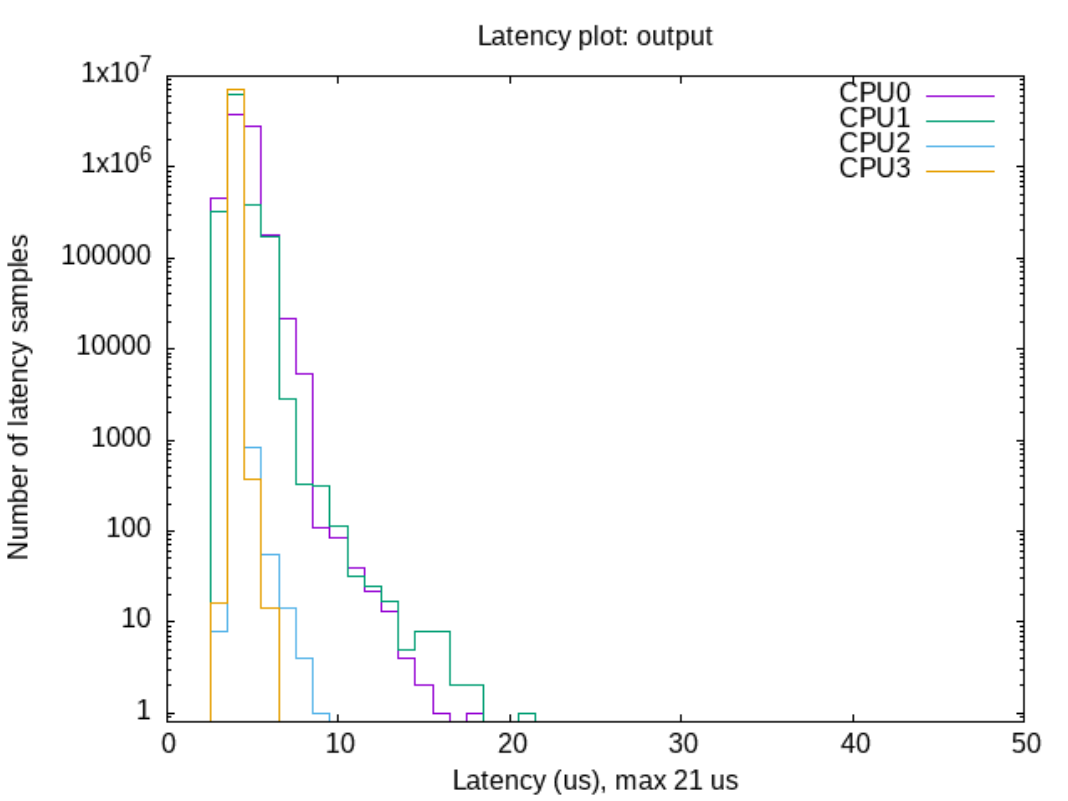
Linux Setup

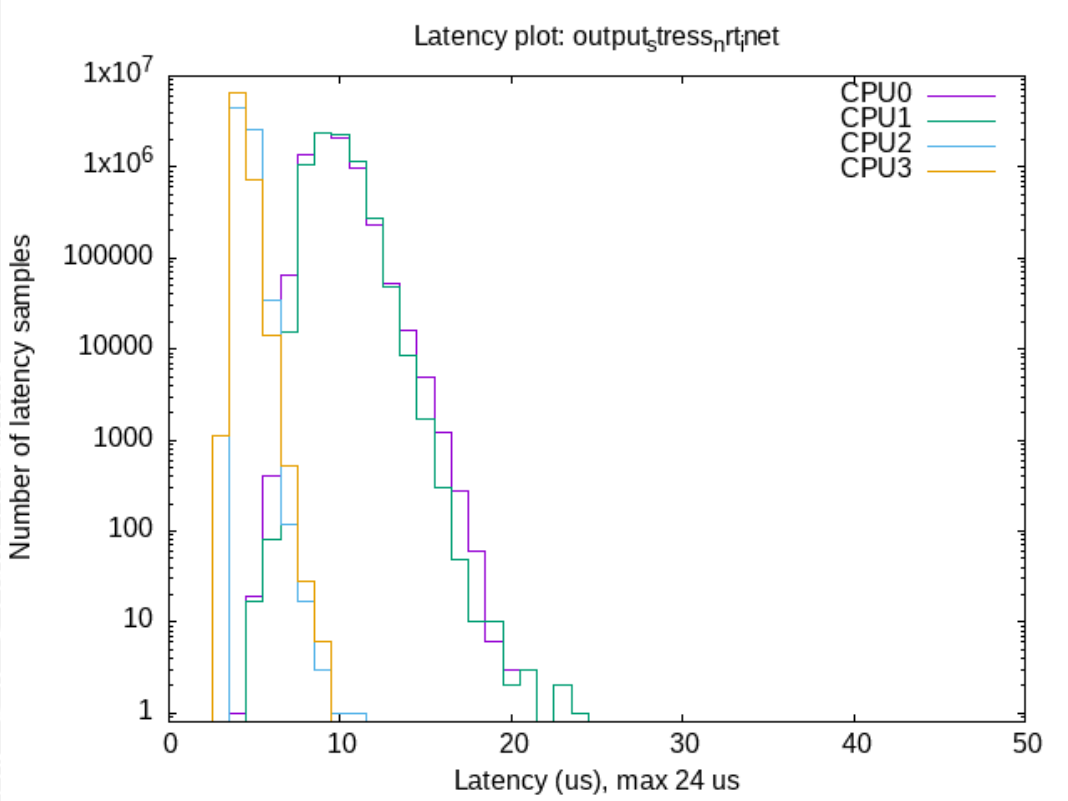
- Laufzeit → cgroup, Prio, Memory
  - `mkdir /dev/cpuset`
  - `mount -t cpuset cpuset /dev/cpuset`
  - `mkdir /dev/cpuset/rt`
  - `cd /dev/cpuset/rt`
  - `echo 2-3 > cpus`
  - `echo 0 > mems`
  - `echo 1 > cpu_exclusive`
  - `echo 1 > mem_exclusive`
  - `echo 0 > sched_load_balance`
- Siehe elbe_files/04_minbase_rt_cpuset.xml

Ein paar Tests - Demo

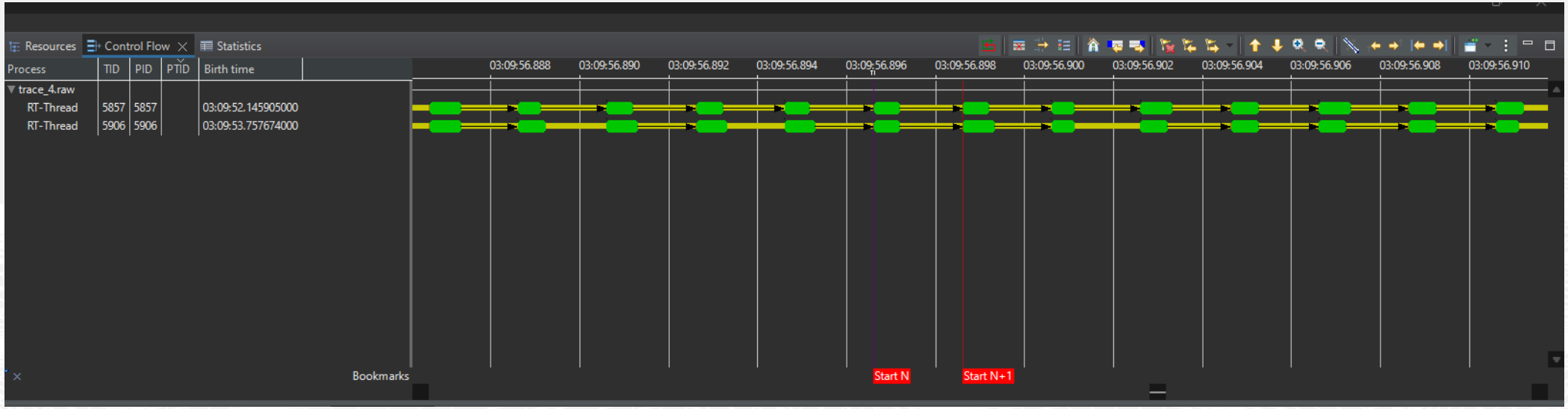
- Cyclictest (mit und ohne cpuset bzw. isolcpus)
- stress-ng (cpu, memory, io, network)

Ergebnis - Scheduling-Latenzen (Arm64 Quad-Core)





Ergebnis - Scheduling-Latenzen (Arm64)



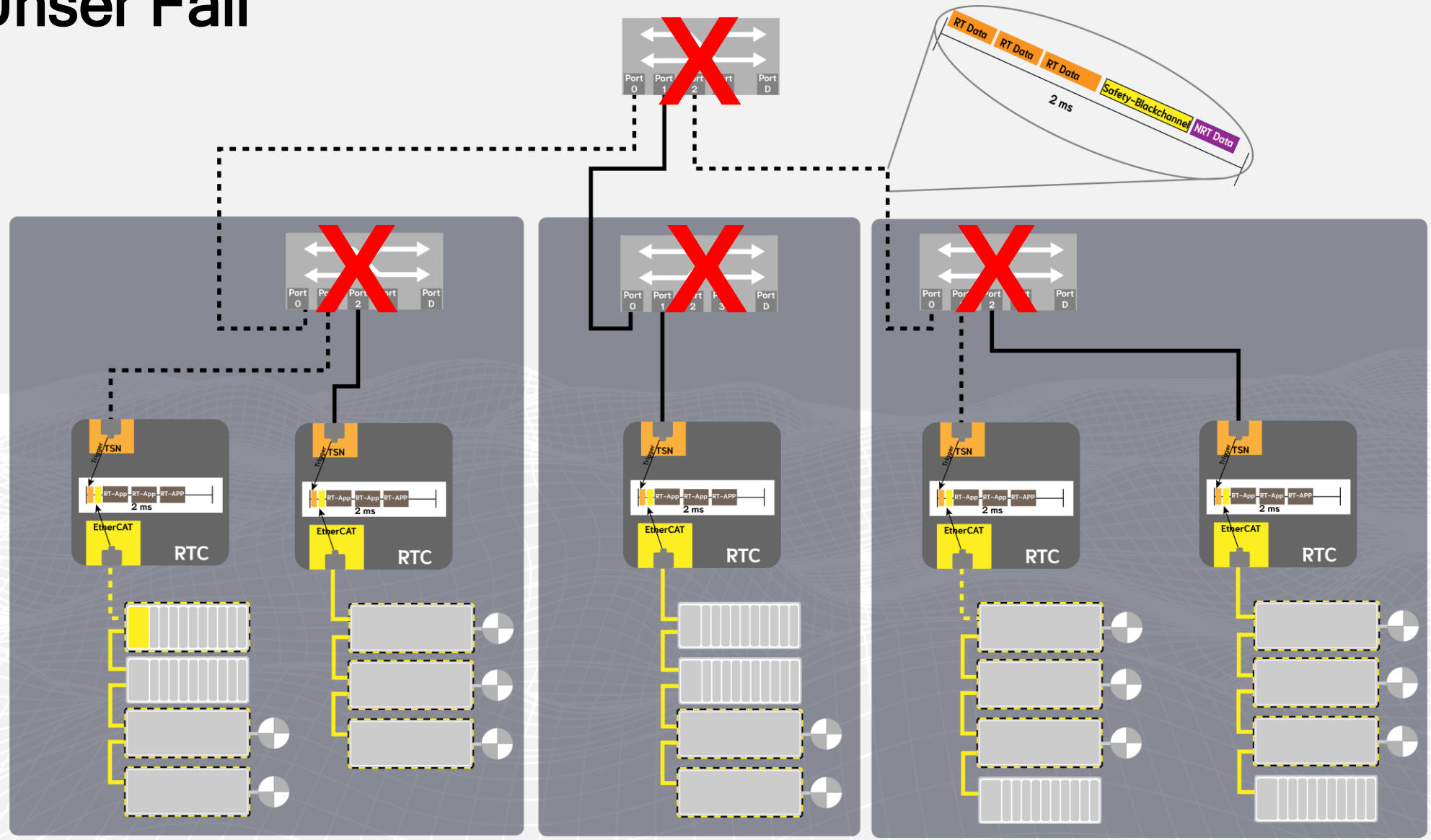
Agenda

- Projektbeispiel
- Bausteine von Echtzeit-Linux
- System Setup
- Applikation (en)
- Entwicklungs-Infrastruktur

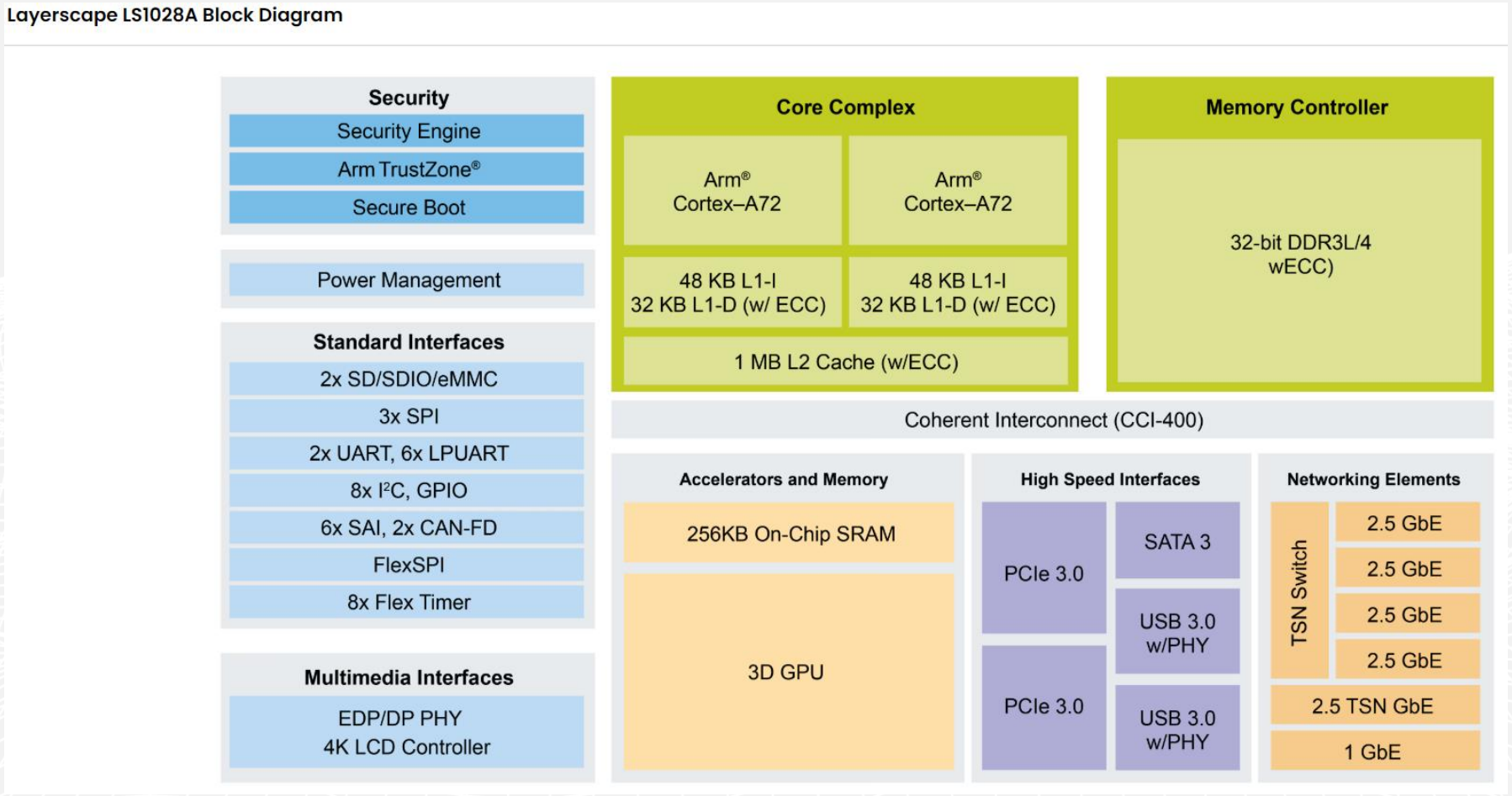
Einschub – Applikation und System

- Fragestellung: Setup in Elbe-XML oder Applikations-Debian-Paket?
- Beispiel: Aktuelles Projekt
- Einrichtung via Applikationspaket
  - Wird mit Applikation installiert und eingerichtet → Keine Abhängigkeit zu System-Setup
  - Ggf. komplexe Änderungen am System. Systemsicht wird ggf. nicht betrachtet.
  - Abhängigkeiten und Konflikte können schön über APT abgebildet werden.

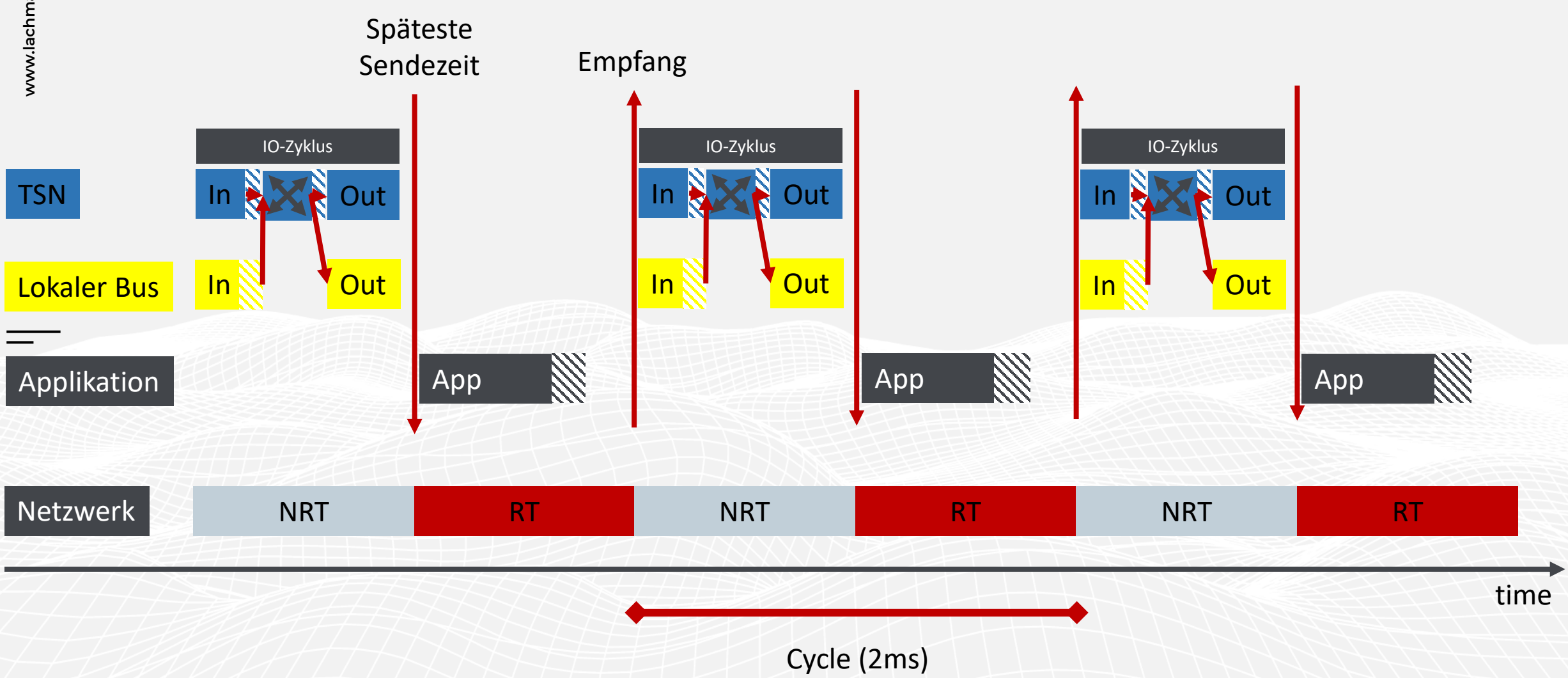
Unser Fall


RT Data
RT Data
RT Data
Safety-Blackchannel
NRT Data
2 ms
Port 0
Port D
TSN
Trigger
RT-App
RT-APP
2 ms
EtherCAT
RTC

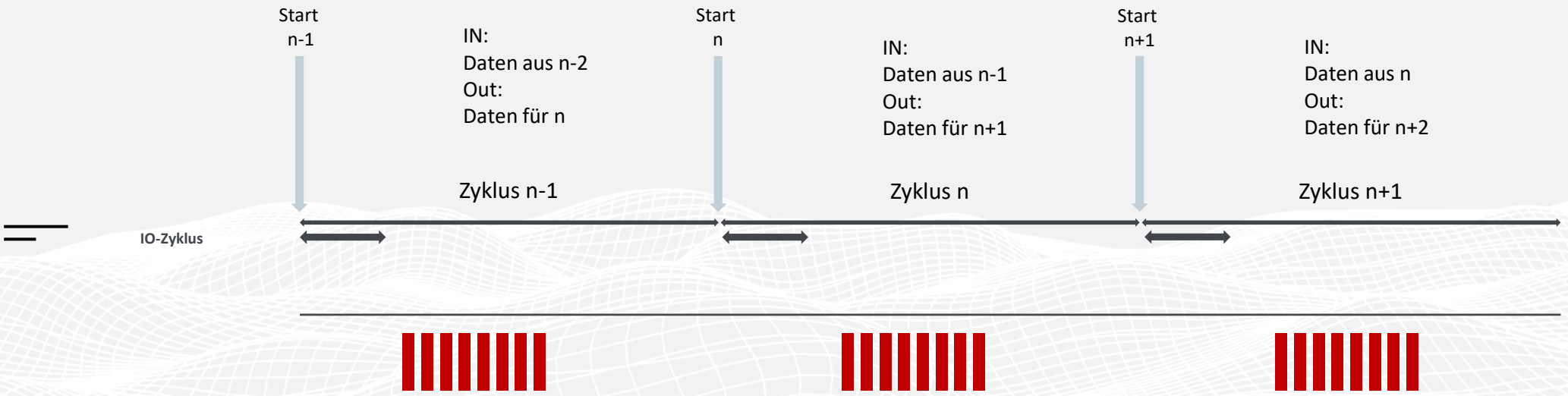
Der Aufbau


Layerscape LS1028A Block Diagram
Security
Security Engine
Arm TrustZone®
Secure Boot
Power Management
Standard Interfaces
2x SD/SDIO/eMMC
3x SPI
2x UART, 6x LPUART
8x I²C, GPIO
6x SAI, 2x CAN-FD
FlexSPI
8x Flex Timer
Multimedia Interfaces
EDP/DP PHY
4K LCD Controller
Core Complex
Arm® Cortex–A72
Arm® Cortex–A72
48 KB L1-I 32 KB L1-D (w/ ECC)
48 KB L1-I 32 KB L1-D (w/ ECC)
1 MB L2 Cache (w/ECC)
Memory Controller
32-bit DDR3L/4 wECC)
Coherent Interconnect (CCI-400)
Accelerators and Memory
256KB On-Chip SRAM
3D GPU
High Speed Interfaces
PCIe 3.0
PCIe 3.0
SATA 3
USB 3.0 w/PHY
USB 3.0 w/PHY
Networking Elements
TSN Switch
2.5 GbE
2.5 GbE
2.5 GbE
2.5 GbE
2.5 TSN GbE
1 GbE

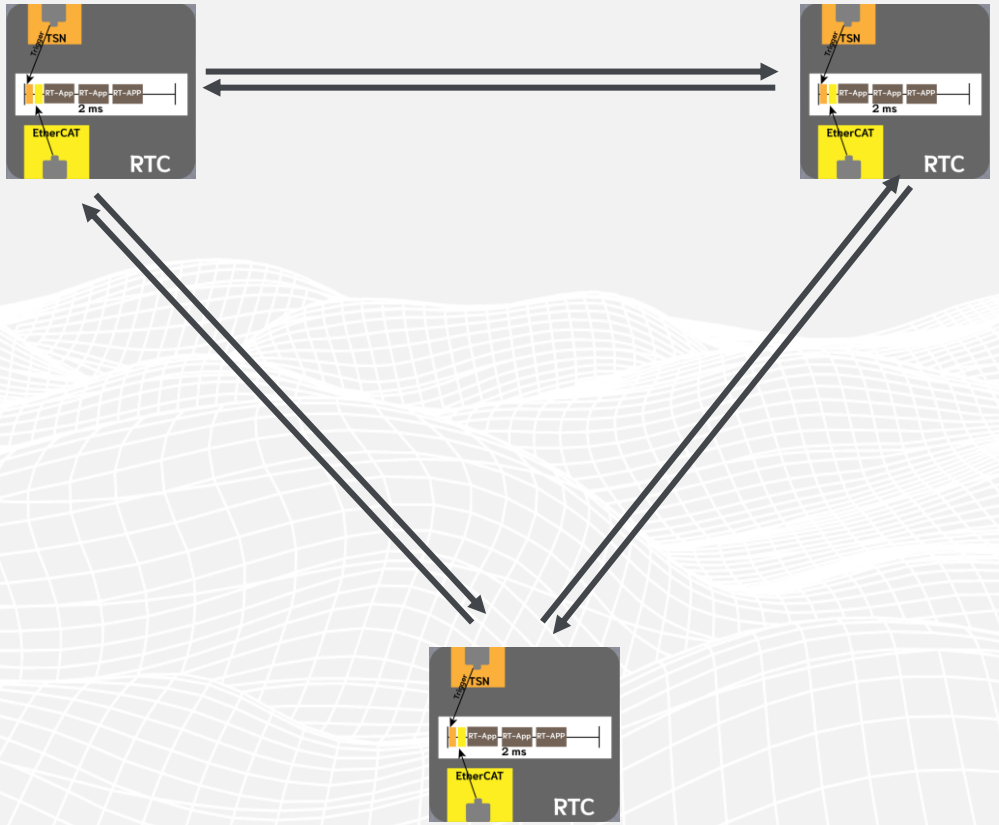
Switched-Endpoint – Perspektive Zyklusablauf



Zyklusdaten - Netzwerksicht



TSN - Schema Streams



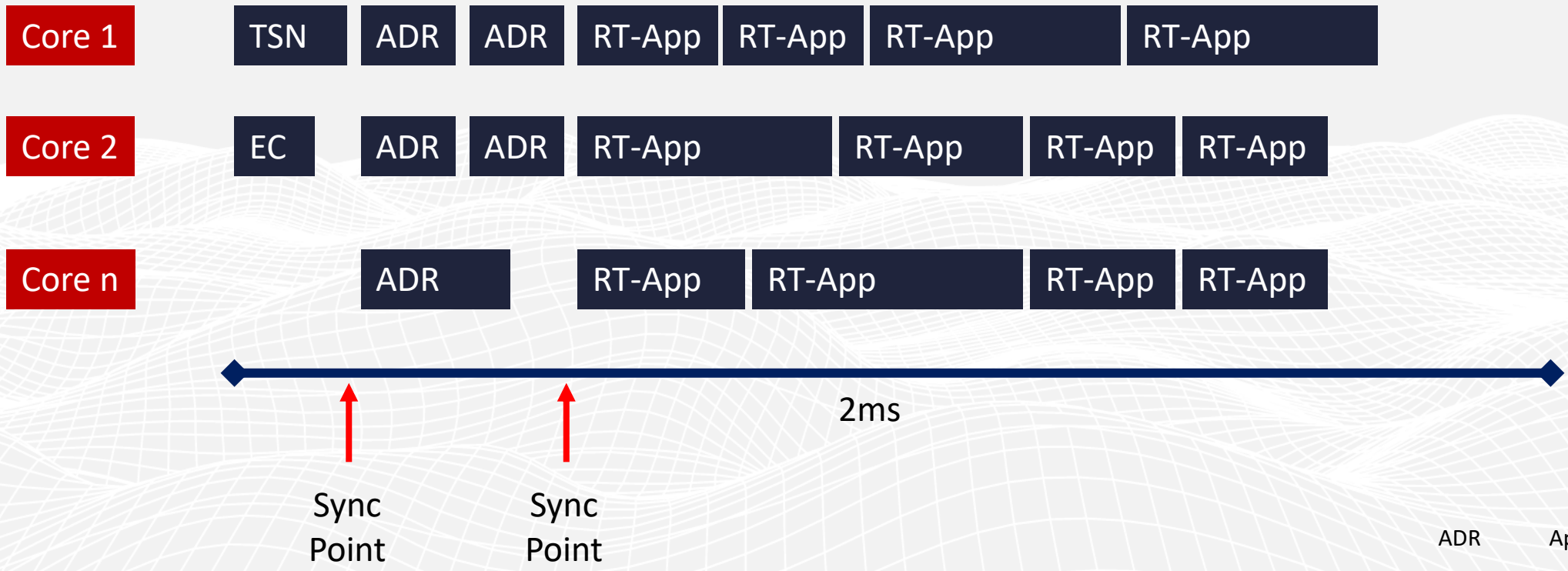
TSN - Was wir brauchen

- Eigenen Device-Tree für Switch-Support
- System-Setup
  - Bridge-Device
  - Qdisc Settings
  - Vlan-Setup
  - PTP-Setup
  - Switch Setup (Gate-Control-List)
- In unserem Fall: System-Setup
- Siehe elbe_files/05_minbase_rt_bridge_setup.xml und
- Siehe elbe_files/06_minbase_rt_bridged_endpoint.xml

Agenda

- Projektbeispiel
- Bausteine von Echtzeit-Linux
- System Setup
- Applikation (en)
- Entwicklungs-Infrastruktur

Echtzeit in diesem Kontext



Generelle Einstellungen

```
int main()
{
    set_cpuset();
    prefault_stack();
    prefault_heap();
    lock_memory();
    setup_scheduler();
    …
}
```

Prefaulting

- Jeder RT-Thread muss den eigenen Stack selbst prefaulten
- Wie viel Heap prefaulten?
  - Speicher-Notwendigkeit vorher messen

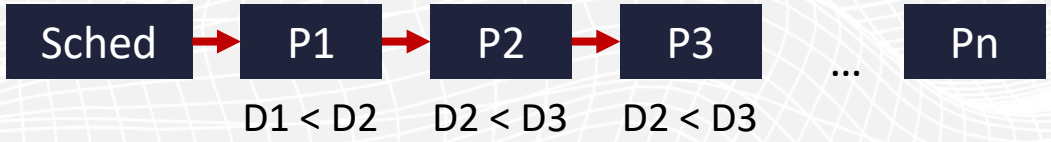
Der Scheduler

- SCHED_DEADLINE
- SCHED_FIFO
- SCHED_RR

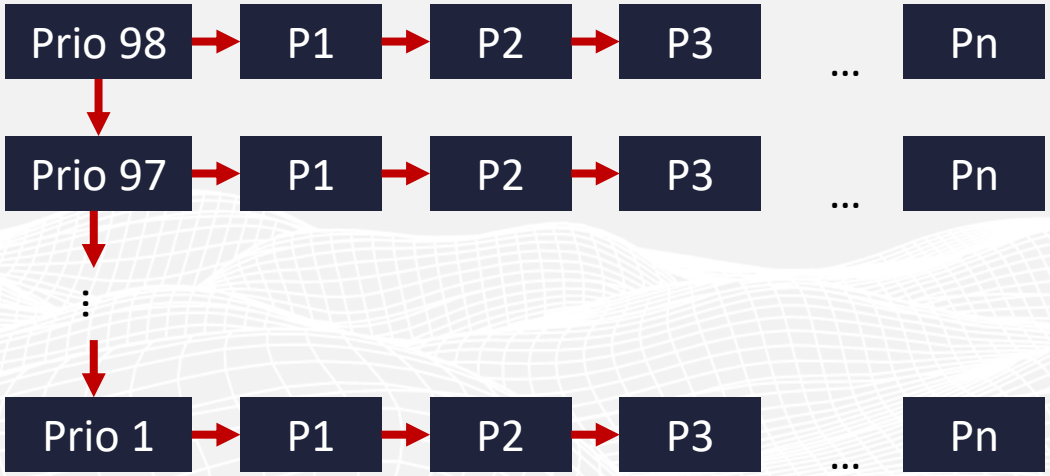
Sched_DEADLINE

CBS

EDF



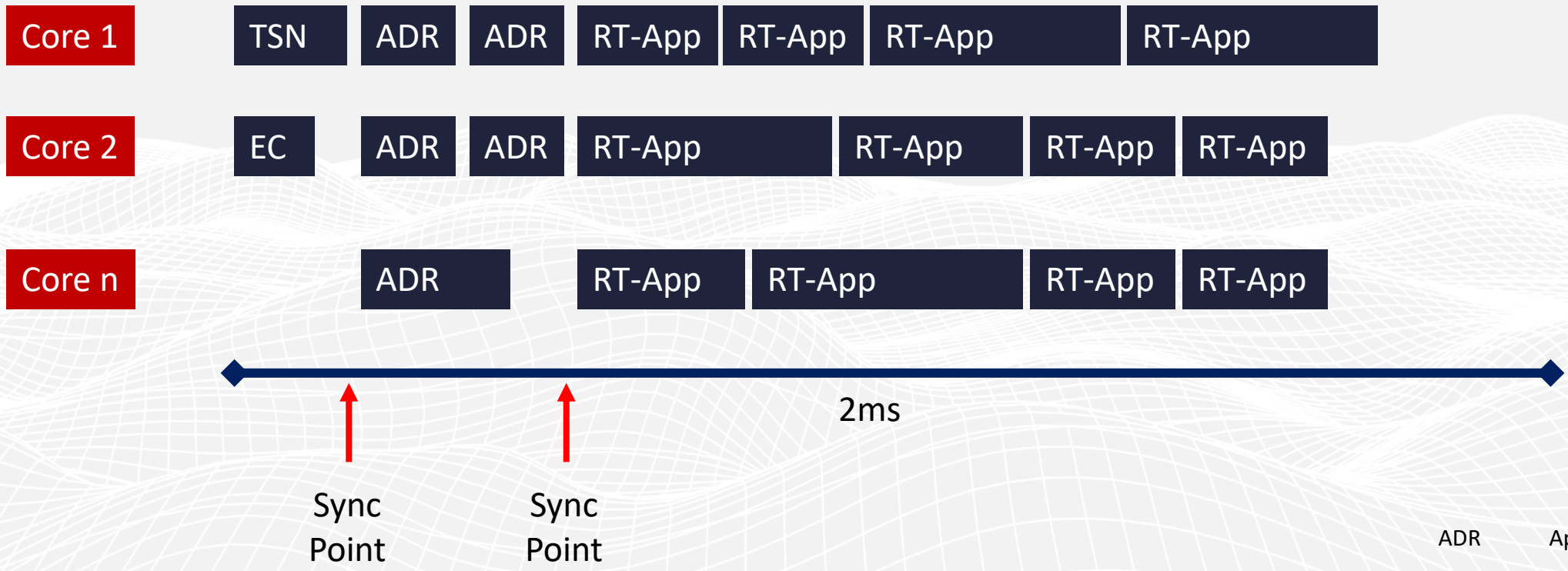
Sched_FIFO



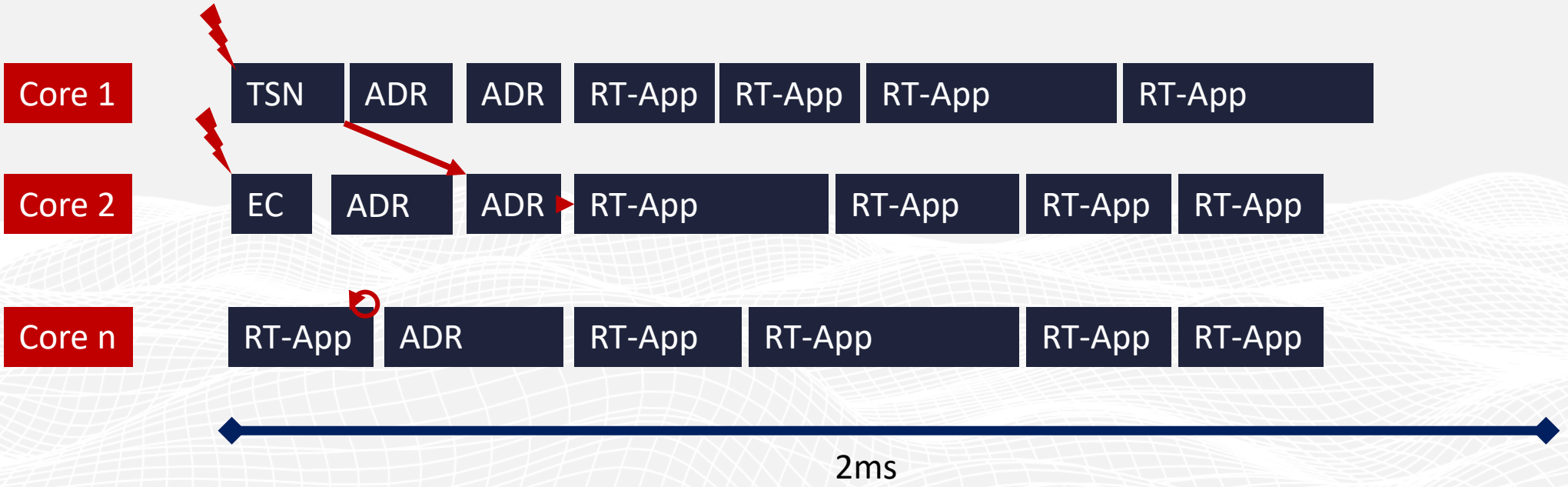
Die Main Loop

```
int main()
{
    setup_scheduler();
    …
    while(1){
        …
        clock_nanosleep(…);
        //Do stuff
    }
}
```

Echtzeit in diesem Kontext



Echtzeit in diesem Kontext



ADR Appl. Driver

RT-APP App oder Thread

Apps und Sync-Points

- Apps oder Threads
- Arten von Apps
  - Zyklisch aufzurufen → Trigger
    - Nanosleep
  - Abhängige (Abfolge, Daten, …) → Trigger
    - IPC
    - Event getriggert
  - Freilaufende → Prio

IPC „Tools“

- Shared Memory
- Shared Condition
- Unix Domain Socket
- Pipes
- ...

Einrichten eines Devcontainers mit Elbe-SDK

- VSCode
- In ese_seminar Verzeichnis wechseln
- F1 drücken
- Build and Reopen in Container wählen
- Anfangen zu Arbeiten
- Demo

www.lachmann-rink.de

LACHMANN & RINK | PARTNER FOR SMART INDUSTRIAL SOLUTIONS

Vielen Dank!

LACHMANN & RINK | PARTNER FOR SMART INDUSTRIAL SOLUTIONS

Ihr Ansprechpartner

Martin Steih

+49 2734 2817-430

martin.steih@lachmann-rink.de

Hommeswiese 129, 57258 Freudenberg | Otto-Hahn-Straße 18-20, 44227 Dortmund

www.lachmann-rink.de | info@lachmann-rink.de