Ressourcen-Management für die Multicore-Mikrocontroller-Auswahl

Die **Anforderungen an Mikrocontroller-gesteuerte Systeme steigen** von Jahr zu Jahr.

Unsere Erwartungen an technische Geräte nach **mehr Komfort**,
erweiterte Funktionalität und **höhere Sicherheit** bei der Anwendung steigen stetig.

Die Mikrocontroller benötigen also immer mehr Rechenleistung zur Bewältigung der an sie gestellten Aufgaben.

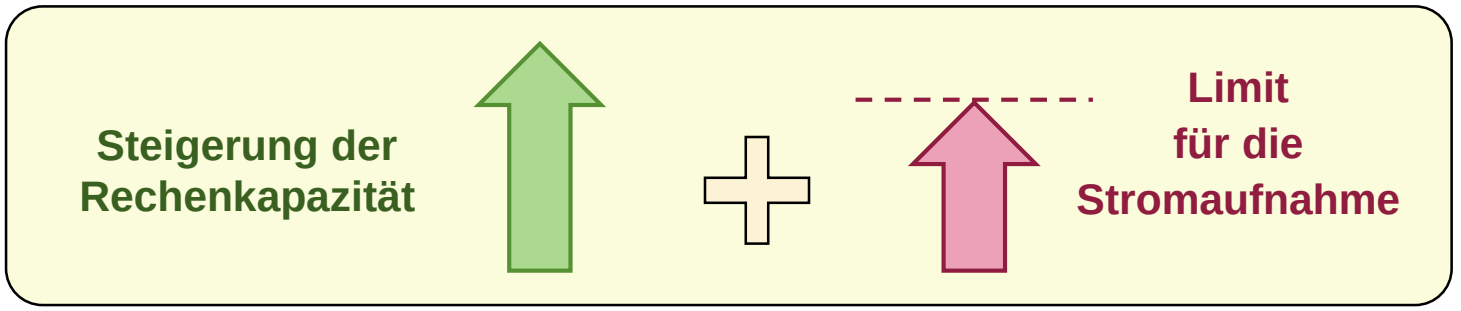
Die **Funktionalitätserweiterungen erhöhen** den **Stromverbrauch** in den Steuerungen.

Der **Spagat zwischen System-Leistungsfähigkeit** und der **Akkulaufzeit** zeigt die Grenzen der Machbarkeit bei batteriebetriebenen Geräten (z.B. Mobiltelefonen) auf.

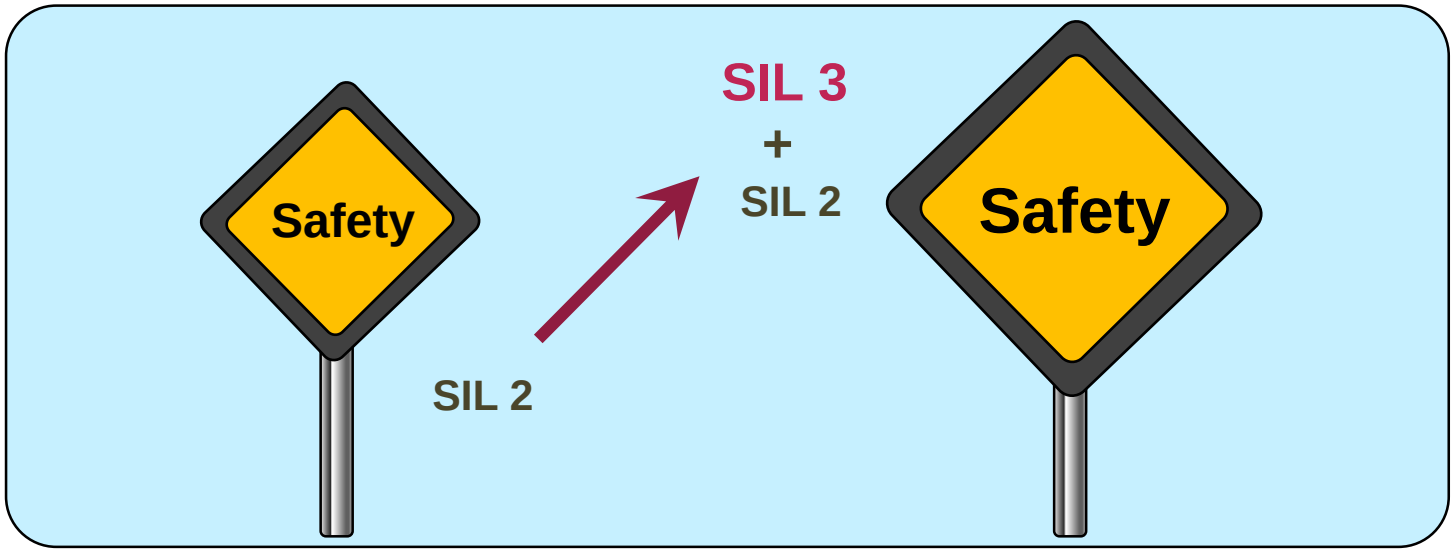
Die für den Betrieb benötigten Akkus können nicht immer mehr Strom liefern, ohne dass das einen Einfluss auf die Größe und das Gewicht der (Handheld-)Geräte hat.

Bussysteme mit **höherer Datentransferleistung** stellen weitere Herausforderungen dar.

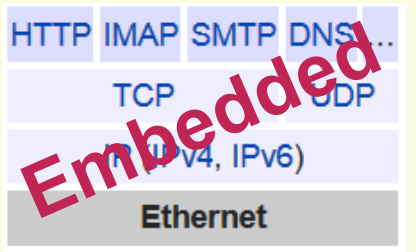
Komplexere Aufgabe



Höhere Safety-Anforderung



Zusätzliche Kommunikations-module



MICRO CONSULT

Mehr MIPs pro Watt

Bei Notebooks hat man den Weg zuerst beschritten, die **Steigerung** der **Rechenleistung** durch **Multicore-CPUs** bei **moderater Taktung** zu bewerkstelligen.

Dieser Lösungsansatz wurde gewählt, damit der Stromverbrauch nicht zu stark durch eine höhere Arbeitsfrequenz bei Singlecore-Bausteinen steigt.

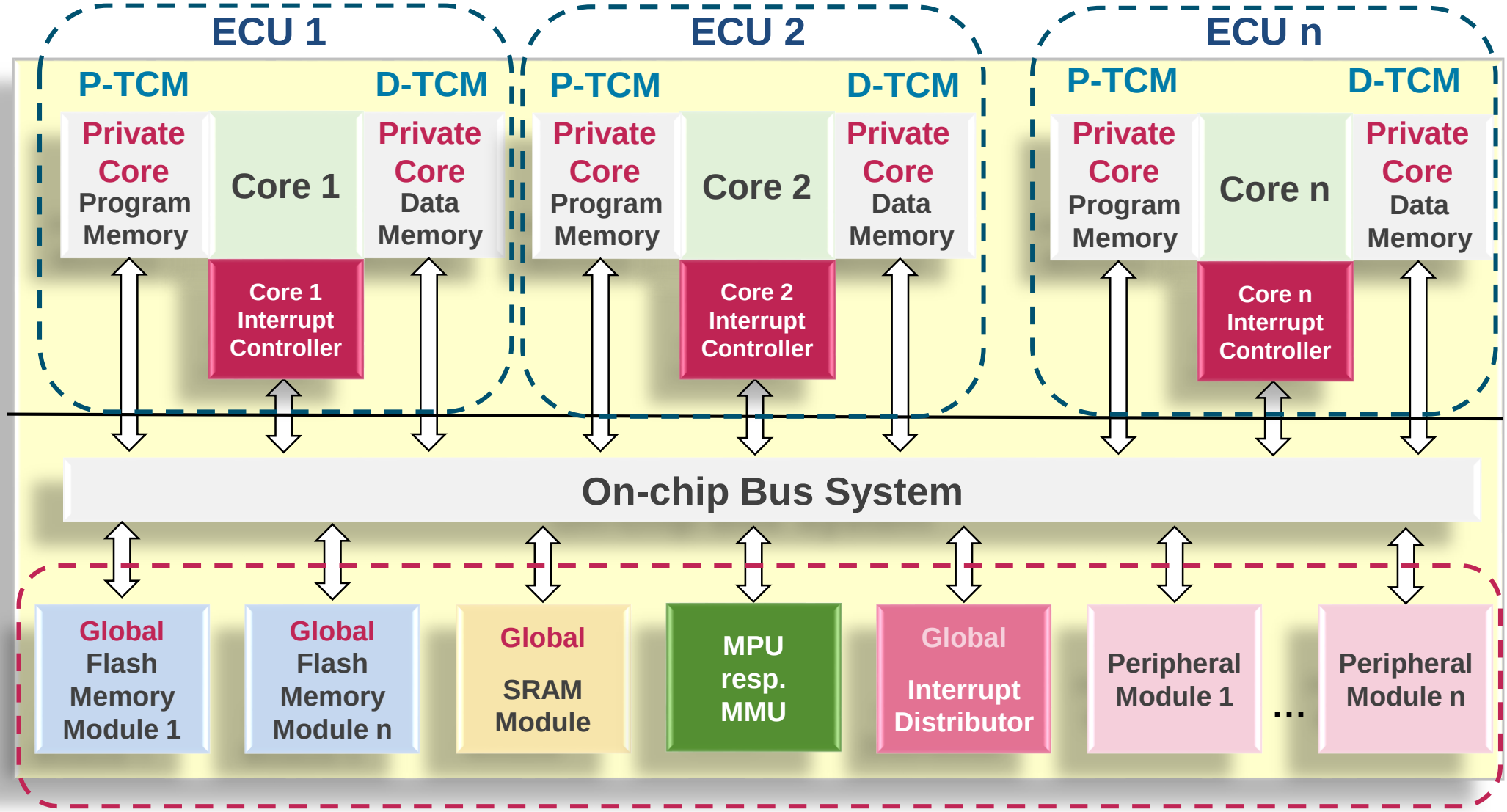
Bei der Mikrocontroller-Entwicklung haben die Multicore-Architekturen mit ein paar Jahren Verzögerung Einzug gehalten. Gleiche Forderungen: **mehr Rechenleistung ohne signifikante Steigerung der Stromaufnahme.**

Automobilindustrie als Vorreiter beim Einsatz der Multicore-Mikrocontroller

Die Entwicklung der **Steuerungssysteme** in der **Automobiltechnik** wird durch zwei Hauptaspekte vorangetrieben:

- die **Einhaltung gesetzlicher Forderungen** bezüglich **Verbrauch** von Treibstoff, resp. die Limitierung des **Schadstoffausstoßes**
- die Forderung nach höherer Leistungsfähigkeit bezüglich der **Kommunikations- und Entertainment-Systeme** im **Auto** und besonders das **autonome Fahren**.

Single-Chip Multicore Design



Shared Resources

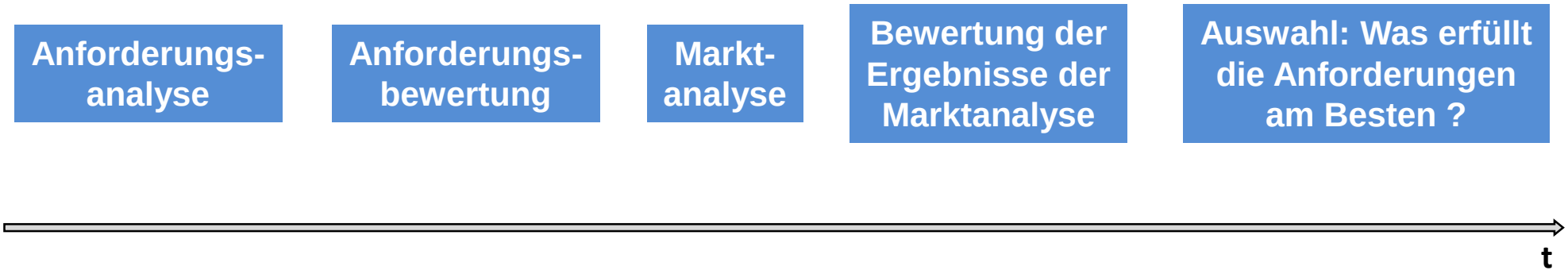
MICRO CONSULT

Schritte zur erfolgreichen Multicore-µC-Auswahl

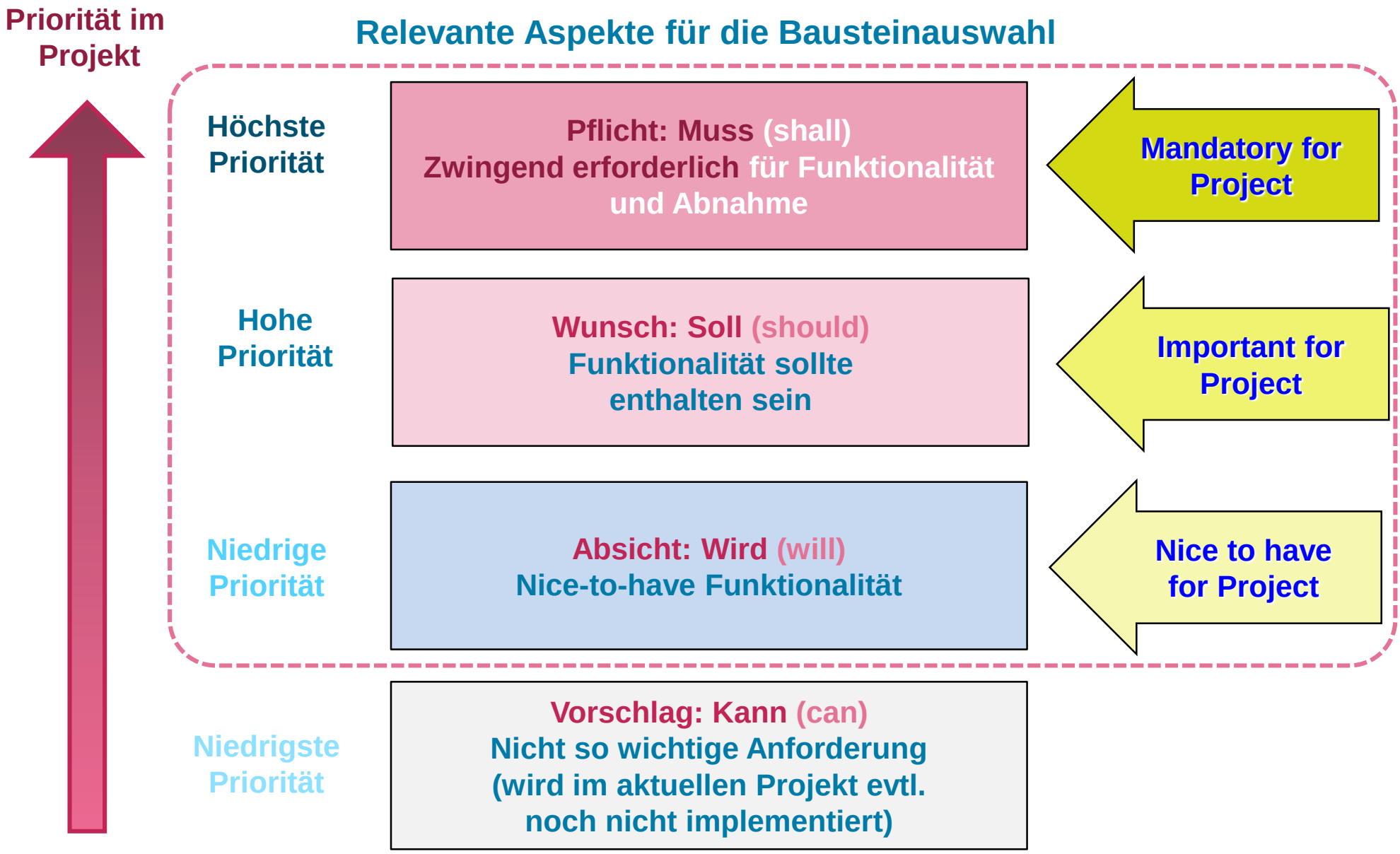
- **Festlegung der Methodik bzw. der Vorgehensweise**

- **Aspekte für die Ressourcen-Identifikation und das Management**
  - **Festlegung des Core-Typs, der benötigter Rechenleistung und der Core-Anzahl**
  - **Analyse der benötigten Datenspeicher-/Programmspeichergröße**
  - **Bestimmung von Realtime-Anforderungen bzw. der Anforderungen an Interrupt- und DMA-Controller**
  - **Bestimmung der geforderten Peripherien und Ports**
  - **Maximal tolerierte Stromaufnahme**
  - **Analyse der On-chip-Kommunikation: Forderungen an Bussysteme**
  - **Analyse und Definition der Safety- und Security-Aspekte**

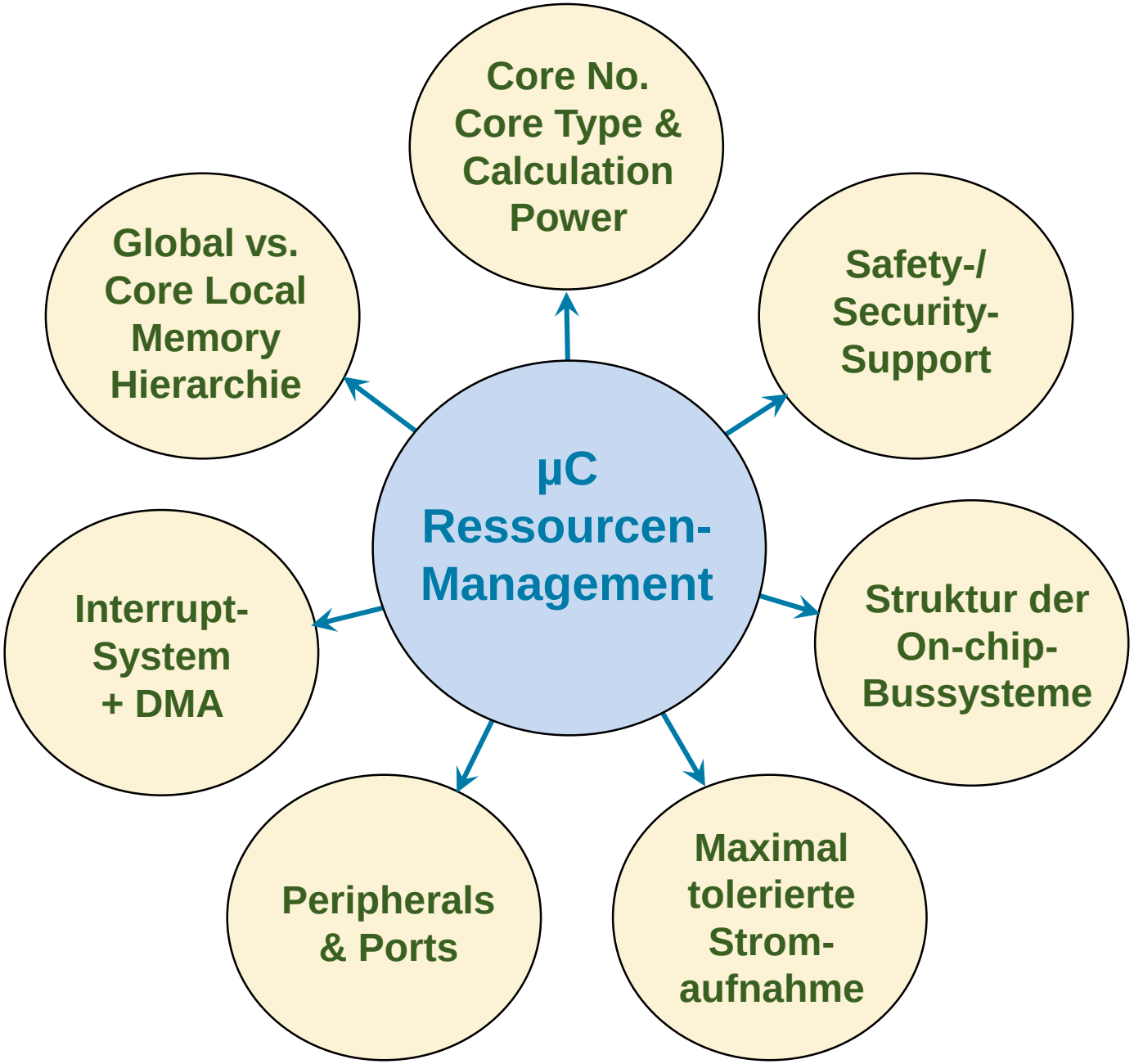
Erfolgreiche Vorgehensweise bei der Multicore-µC-Auswahl



- **Identifikation** und **Dokumentation** der **Anforderungen** für
  - **Software**
  - **Hardware**
- **Bewertung** der **Anforderungen** nach Prioritäten im Projekt:
  - **Höchste Priorität**
  - **Hohe Priorität**
  - **Niedrige Priorität**
  - **Niedrigste Priorität**
- **Marktanalyse** - Welchen Support gibt es für das **Projekt**:
  - **Bausteine: Typ** und **Second-Source**
  - **Tools und Methoden**
  - **Softwaretools, Libraries**
  - **Boards**
  - **Debug und Testsupport**
  - **Hersteller-Support**
  - **Verfügbare Trainings sowie Coaching-Support**
- **Analyse** – Existierendes **Know-how** und **SW** (aus früheren Projekten)
- **Bewertung** der **Aspekte** für den **Projekt-Support**
- **Baustein-Auswahl**


Priorität im Projekt
Relevante Aspekte für die Bausteinauswahl
Höchste Priorität
Pflicht: Muss (shall)
Zwingend erforderlich für Funktionalität und Abnahme
Mandatory for Project
Hohe Priorität
Wunsch: Soll (should)
Funktionalität sollte enthalten sein
Important for Project
Niedrige Priorität
Absicht: Wird (will)
Nice-to-have Funktionalität
Nice to have for Project
Niedrigste Priorität
Vorschlag: Kann (can)
Nicht so wichtige Anforderung (wird im aktuellen Projekt evtl. noch nicht implementiert)

Die 7 wichtigsten Aspekte für das Ressourcen-Management in Multicore-µC



MICRO CONSULT

Die Software-Migration von Singlecore nach Multicore ist ganz einfach ?!

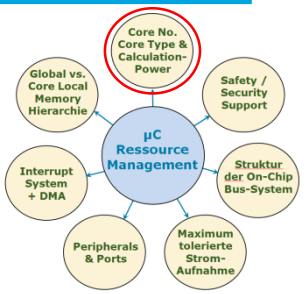
Verarbeiten wir vorhandene Software auf mehreren Cores parallel.
Ein Core entspricht der **Rechenleistung x**, dann haben wir bei einem **Dualcore** also **zweimal Rechenleistung x**.

Effektiv ergibt sich aber eine **Leistungssteigerung** von nur ca. **50 % oder weniger**.
Das ist sehr enttäuschend. Woran kann das liegen?

Die **Singlecore-Softwarearchitektur** ist in der Regel nicht geeignet für die parallele Ausführung/Bearbeitung auf einem Multicore-System.

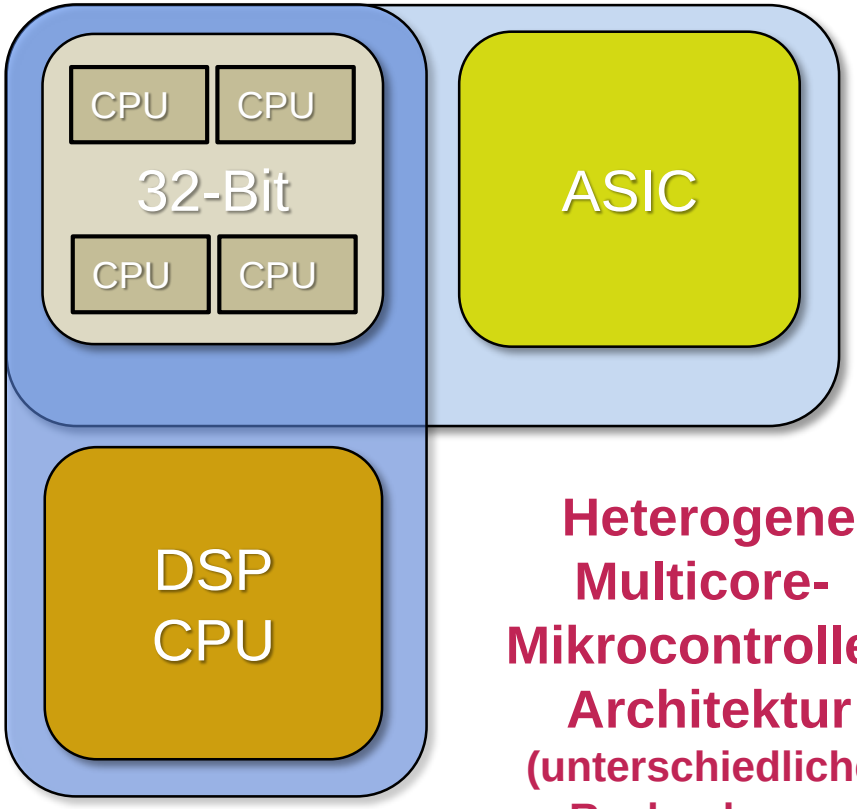
Folgende **Herausforderungen** für einen **Software-Reuse** sind zu bewältigen:

- Singlecore-Software wird sequenziell ausgeführt,
  Multicore-Software benötigt Synchronisationspunkte für ein gleiches Resultat.
- Ein Core greift immer nur auf eine Ressource zu (Datenspeicher/Peripherie/etc.), eine Multicore-Architektur kann gleichzeitig auf eine gemeinsam verwendete Ressource zugreifen; zur Vermeidung inkonsistenter Datenzustände werden Zugriffs-Koordination (Semaphore) und Speicher-Zugriffsschutz (MPU) benötigt.
- Ein Rechenkern benutzt allein ein Bussystem. Mehrere parallel arbeitende Rechenkerne können sich bei Buszugriffen in die Quere kommen und damit hohe Zugriffsverzögerungen auf Ressourcen hervorrufen.



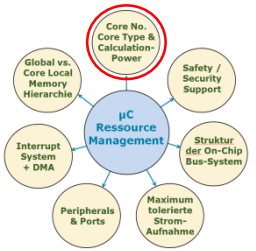
ASIC

Multicore-Microcontroller



DSP

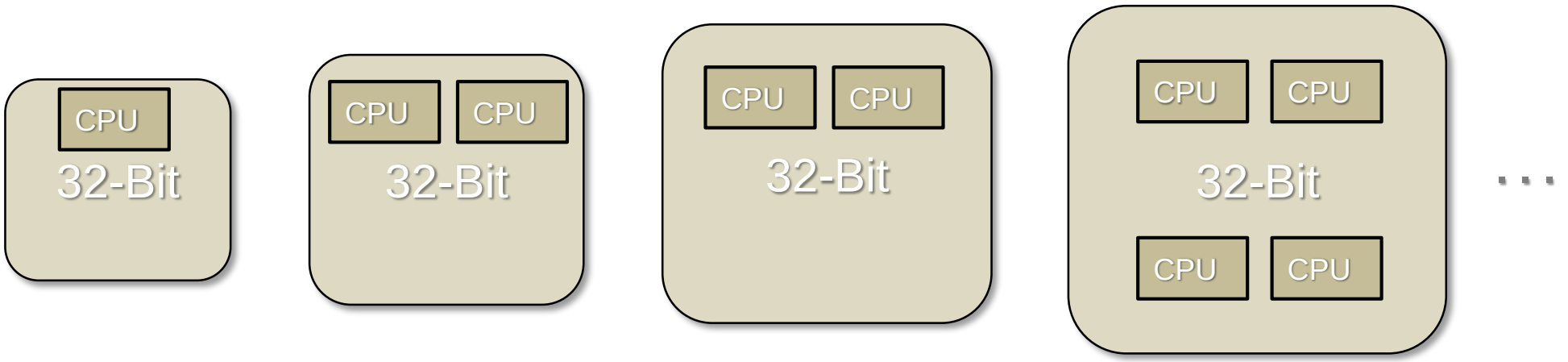
Heterogene Multicore-Mikrocontroller-Architektur
(unterschiedliche Rechenkerne in einem Chip)



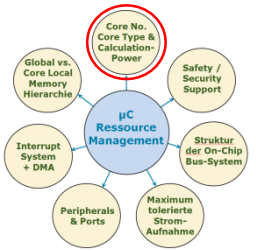
Skalierbare Mikrocontroller-Designs

Unterschiedliche

- Singlecore / Multicore
- Speichergrößen
- Gehäusegröße (Package)
- Peripherie-Implementierungen



Homogene Multicore-Mikrocontroller-Architektur



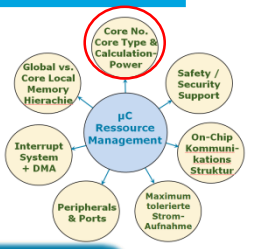
Aspekte für die Laufzeitüberwachung

Gibt es **Laufzeitüberwachungs-Tools** für die eingesetzte **Architektur?**

- **Messung von Task-Laufzeitenzeiten** und **Task-Switch-Zeiten**
- **Laufzeitüberwachung**
- **Messung** von **Blockierzeiten** bei **Verwendung gemeinsamer Ressourcen**

Stehen in der **Architektur Performance-Counter** zur Verfügung zur Messung von:

- **Taktzeiten** der **CPUs**
- **Counter** für **ausgeführte Befehle** (Instruction Counter)
- **Hit-/Miss-Counter** für die **Caches**
- **Bus-Übertragungszeiten?**



Applikation 1
Datenaufnahme

Applikation 2

Applikation 3

Applikation 4

Prozess

Prozess

Prozess

Prozess

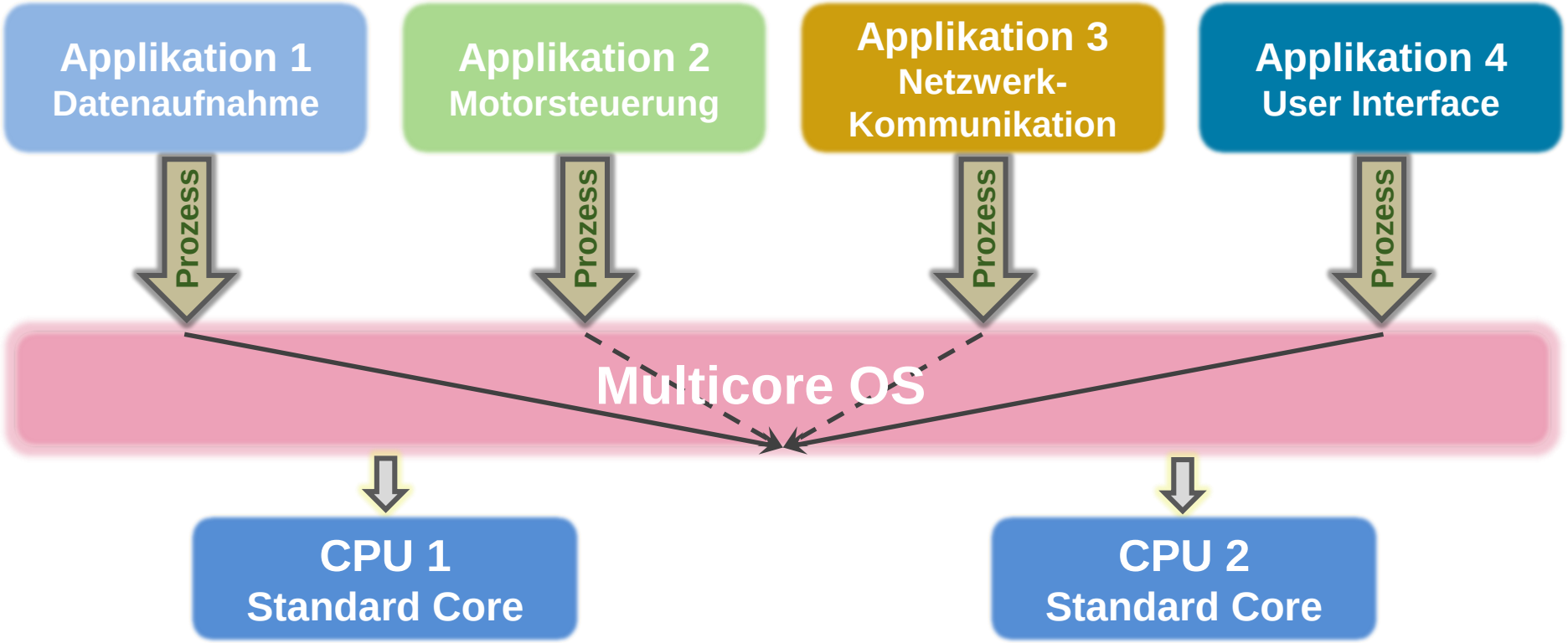
Singlecore OS

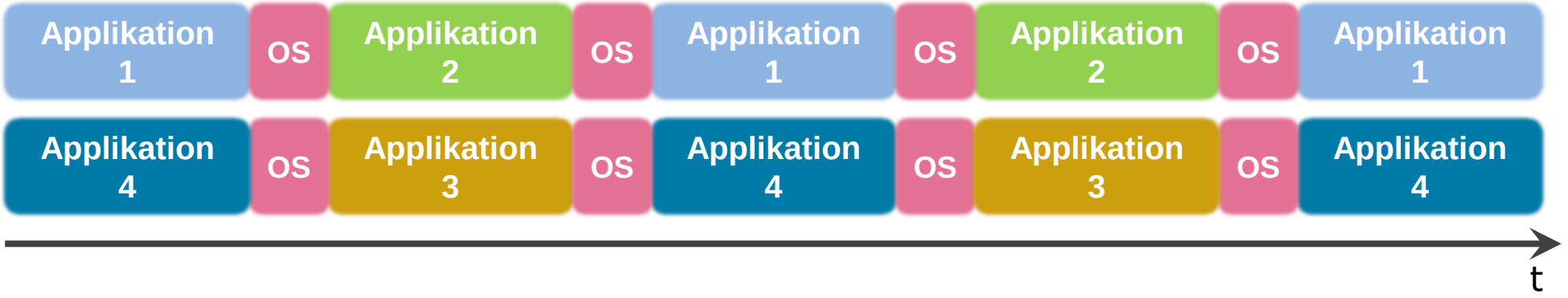
CPU 1
Standard Core

Applikation 1 | OS | Applikation 2 | OS | Applikation 3 | OS | Applikation 4 | OS | Applikation 1

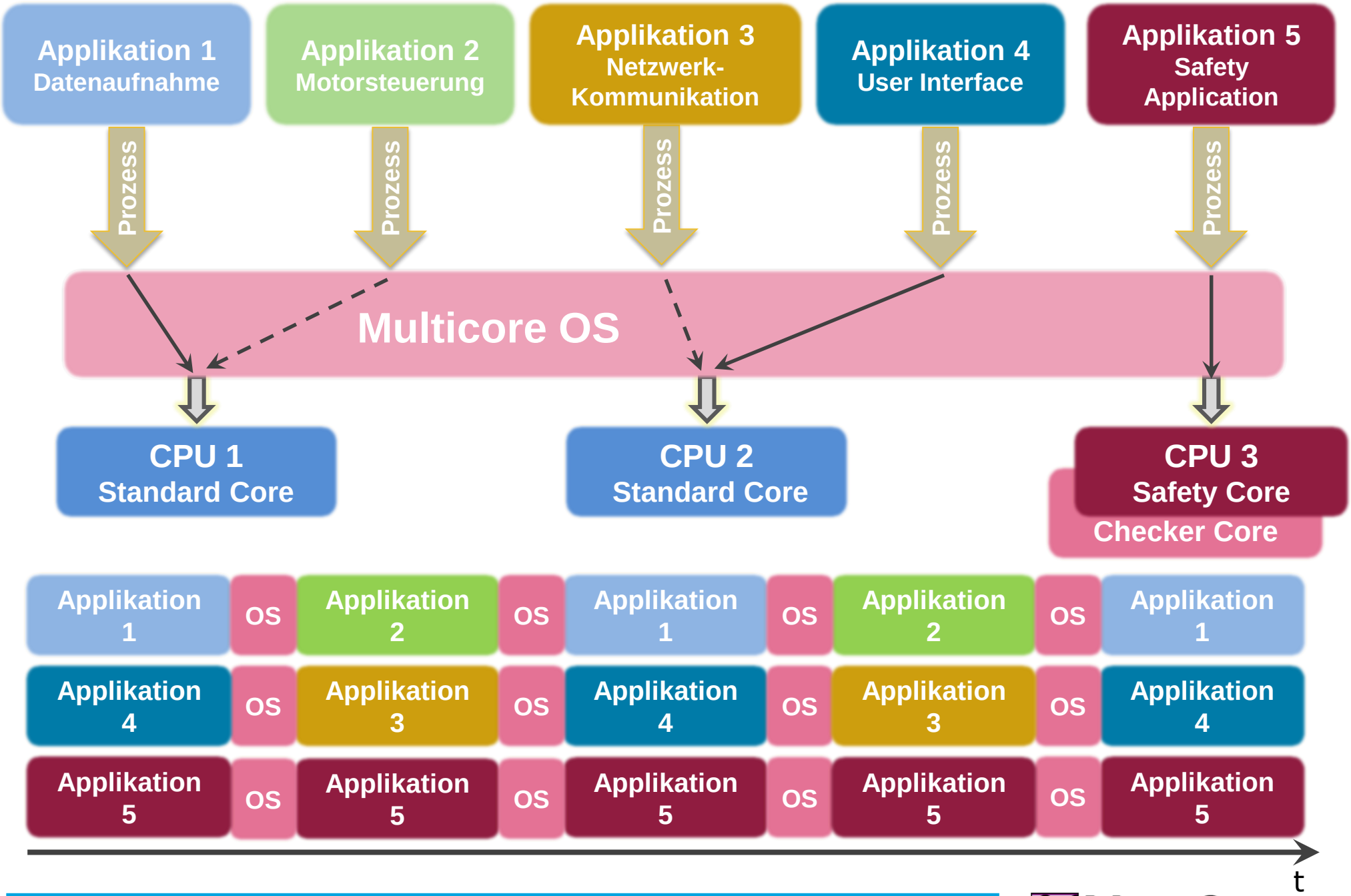
t

MICRO CONSULT

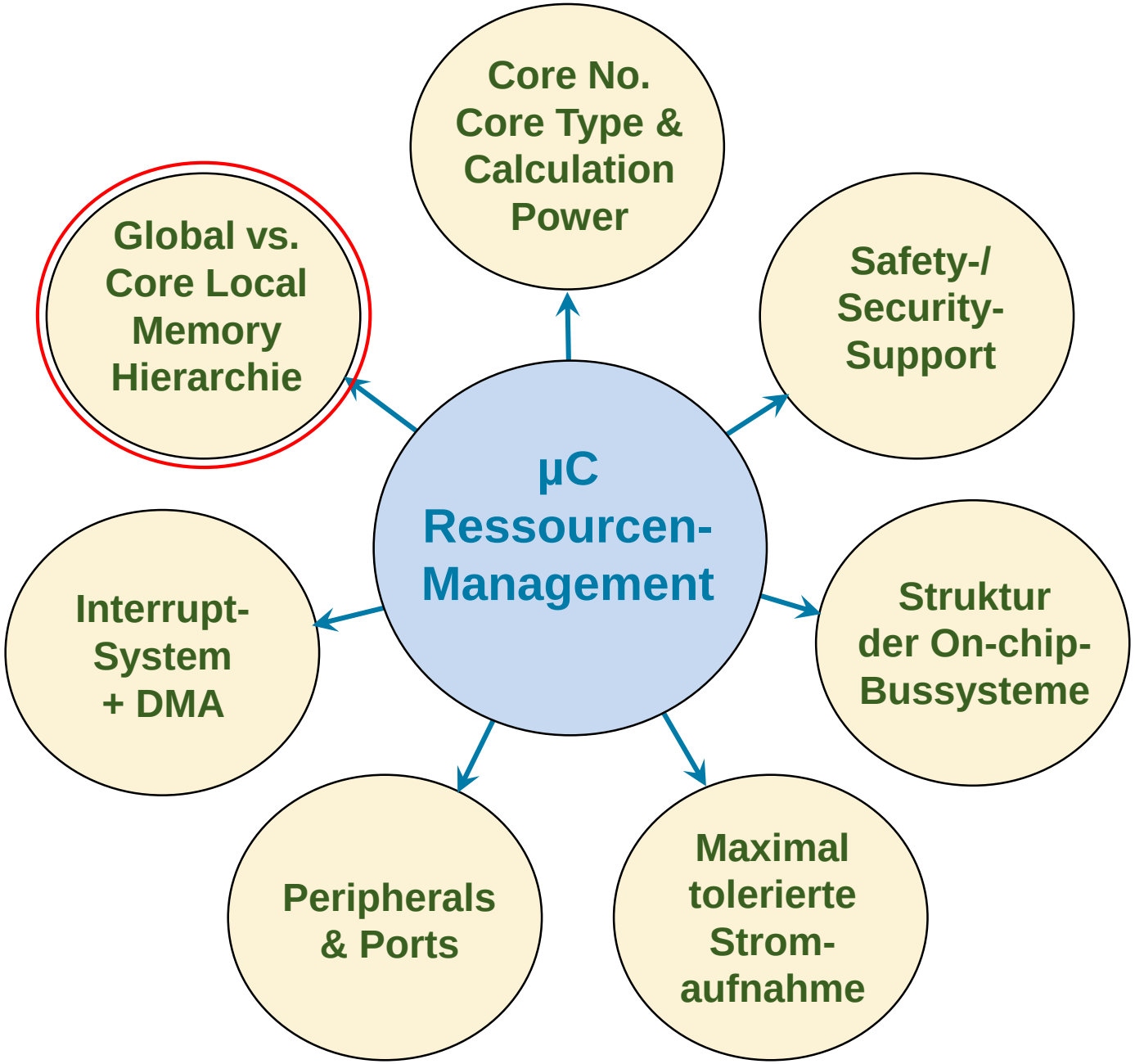

Applikation 1
Datenaufnahme
Applikation 2
Motorsteuerung
Applikation 3
Netzwerk-
Kommunikation
Applikation 4
User Interface
Prozess
Prozess
Prozess
Prozess
Multicore OS
CPU 1
Standard Core
CPU 2
Standard Core


Applikation 1
OS
Applikation 2
OS
Applikation 1
OS
Applikation 2
OS
Applikation 1
Applikation 4
OS
Applikation 3
OS
Applikation 4
OS
Applikation 3
OS
Applikation 4
t

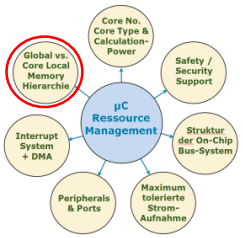
Applikationssoftware-Verteilung bei Multicore Standard- und Safety-SW



MICRO CONSULT


Core No.
Core Type &
Calculation
Power
Global vs.
Core Local
Memory
Hierarchie
Safety-/
Security-
Support
µC
Ressourcen-
Management
Interrupt-
System
+ DMA
Struktur
der On-chip-
Bussysteme
Peripherals
& Ports
Maximal
tolerierte
Strom-
aufnahme

Mikrocontroller-Speicherressourcen – Core-private und globale Speicher



High Speed Core1-lokale Speicher-Ressourcen

High Speed Core2-lokale Speicher-Ressourcen

Core 1

Core-privater Code-Speicher SRAM/ Cache

CPU 1

Core-privater Daten-Speicher SRAM/ Cache

Core 2

Core-privater Code-Speicher SRAM/ Cache

CPU 2

Core-privater Daten-Speicher SRAM/ Cache

Bussystem (z.B. Crossbar-Switch)

Program Flash 1

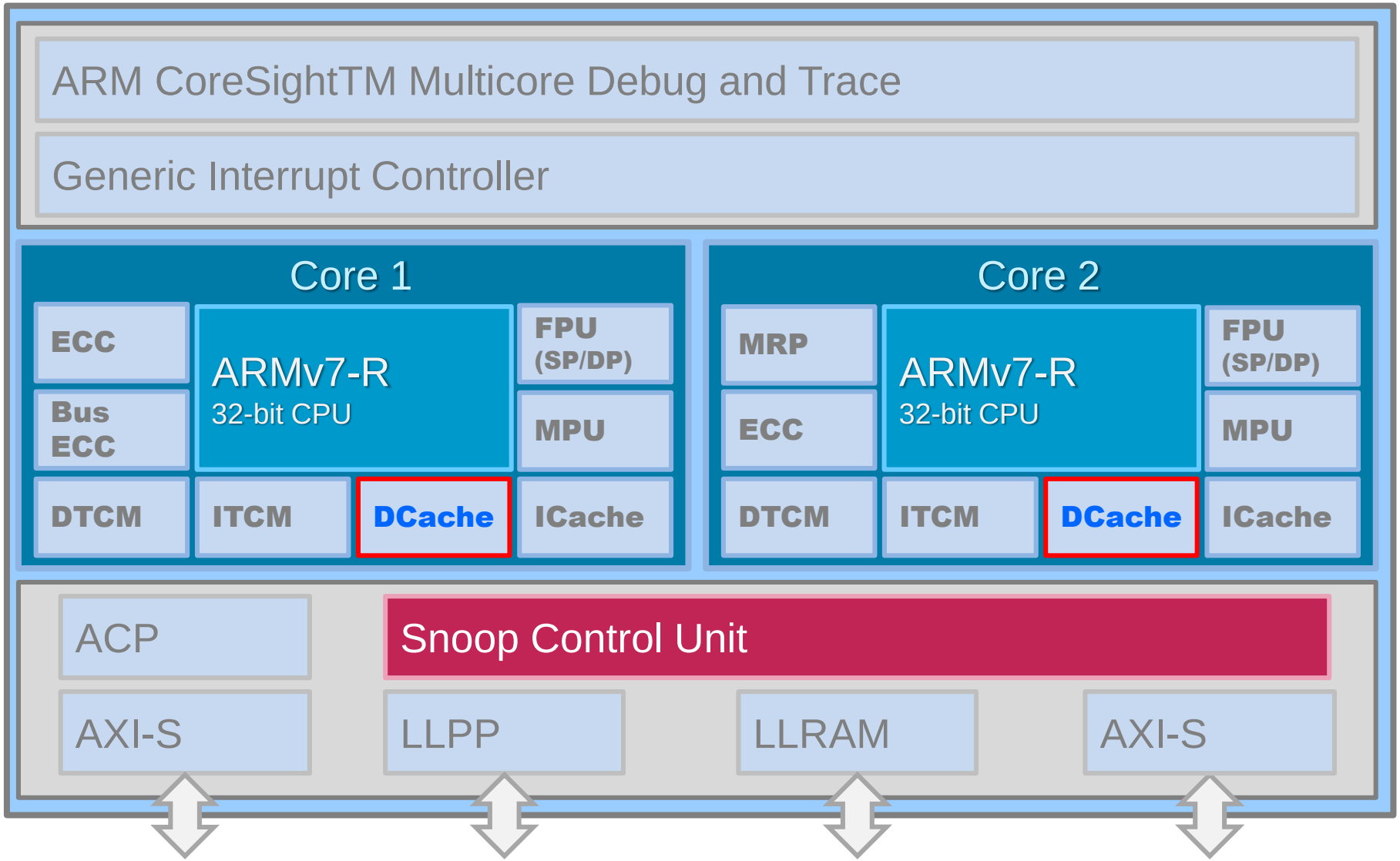
Program Flash 2

Data Flash

Globales SRAM

Globale Speicher-Ressourcen

MICRO CONSULT

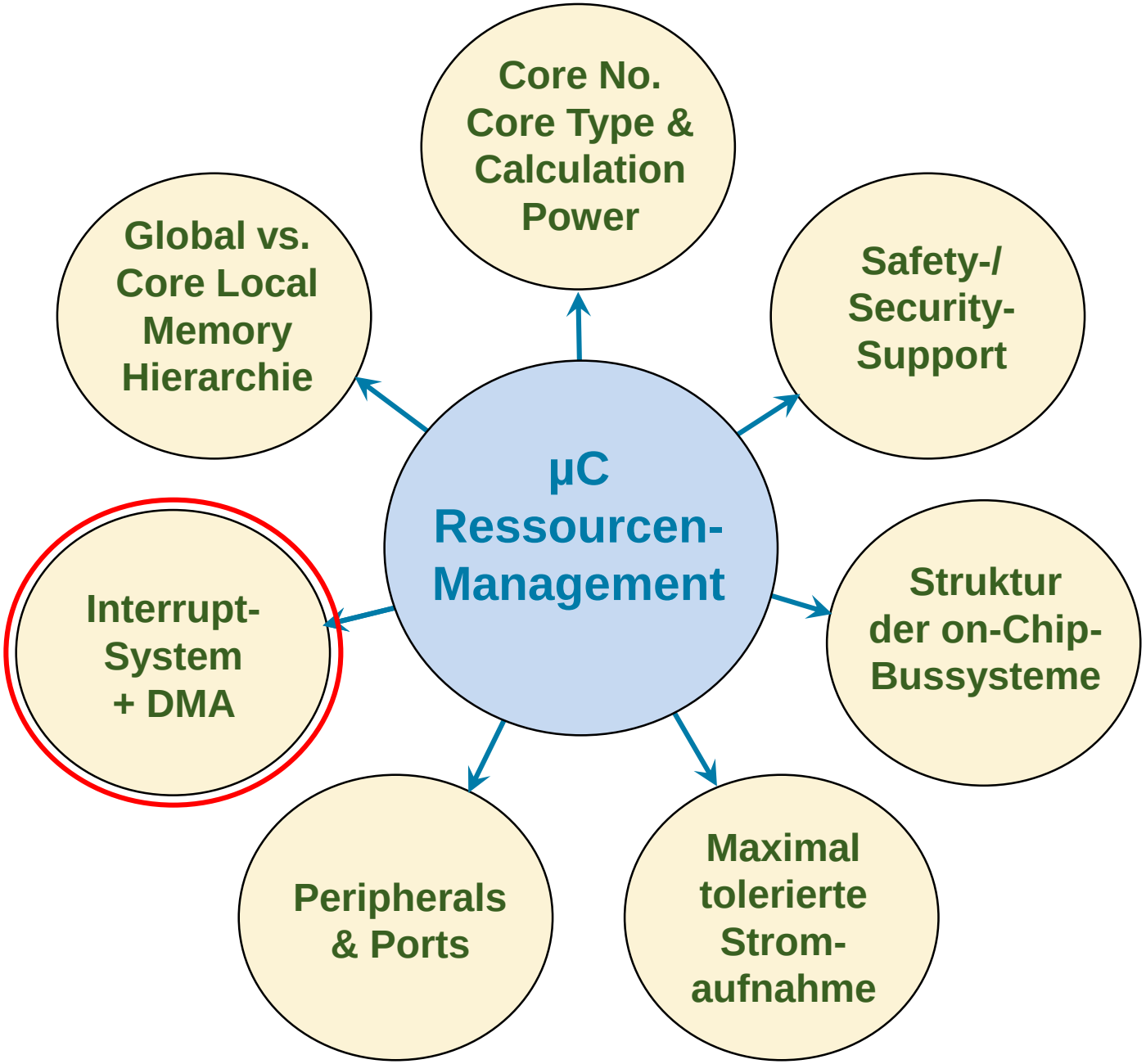


ECC:	Error Correction Code
Bus ECC:	Bus Error Correction Code
DTCM:	Data Tightly Coupled Memory
ITCM:	Instruction Tightly Coupled Mem.
FPU:	Floating Point Unit

MPU:	Memory Protection Unit
ICache:	Instruction Cache
DCache:	Data Cache
AXI-M:	64-bit AMBA® AXI™ Bus Master
AXI-S:	64-bit AMBA® AXI™ Bus Slave

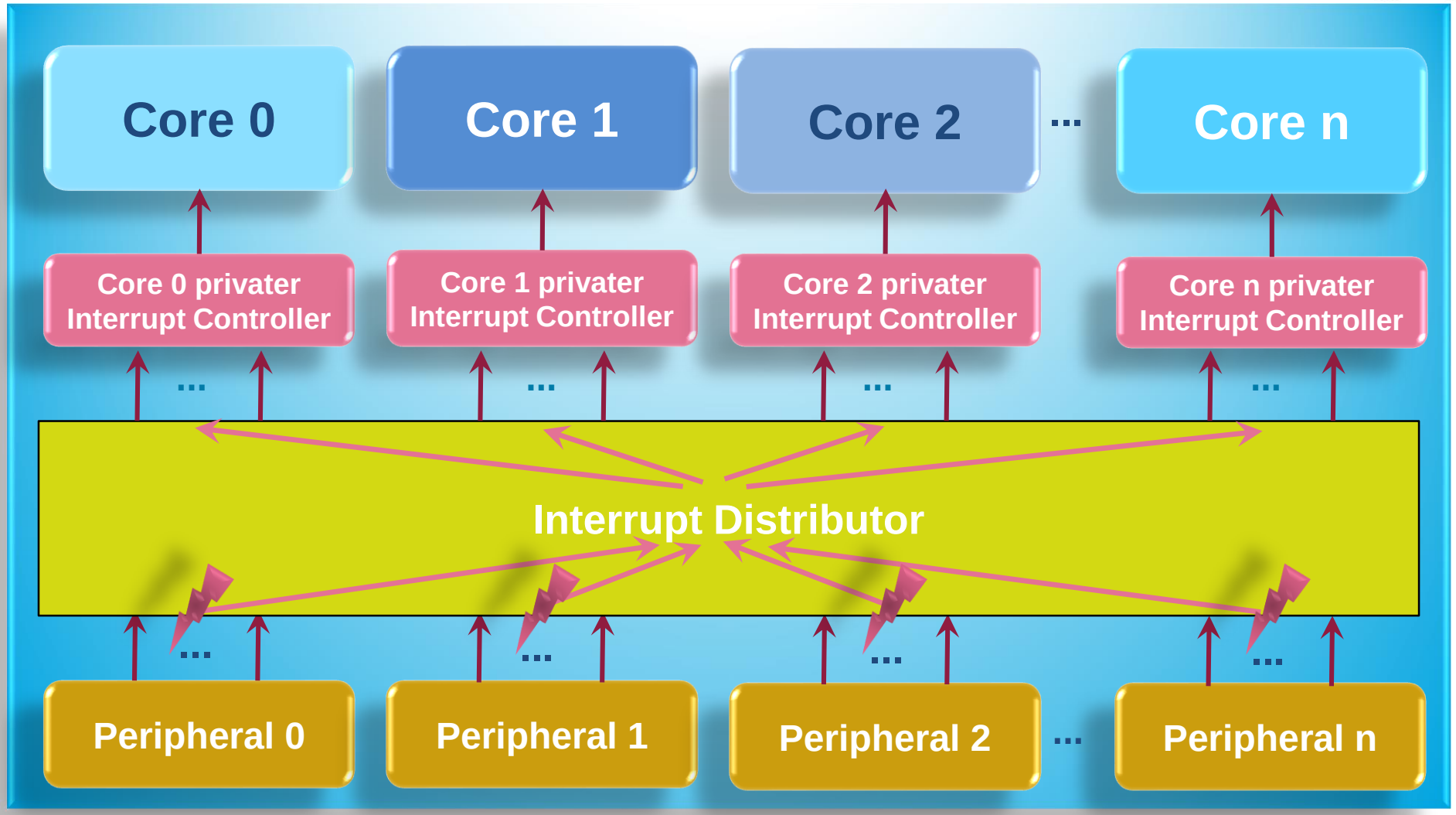
ACP:	Accelerator Coherency Port
LLPP:	Low Latency Peripheral Port
LLRAM:	Low Latency Peripheral Port

Source: www.arm.com

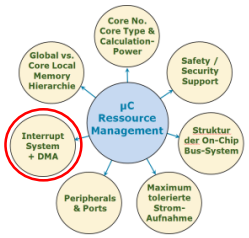

Core No.
Core Type &
Calculation
Power
Global vs.
Core Local
Memory
Hierarchie
Safety-/
Security-
Support
µC
Ressourcen-
Management
Interrupt-
System
+ DMA
Struktur
der on-Chip-
Bussysteme
Peripherals
& Ports
Maximal
tolerierte
Strom-
aufnahme

Jeder Core hat einen eigenen Interrupt-Controller.

Realtime-Peripherie-Ereignisse sind über einen **Interrupt-Distributor** (Interrupt-Router) auf einen **Interrupt-Controller zu verknüpfen**.



MICRO CONSULT



DMA Move

DMA Quelle

Data Source 1

Data Source 2

Data Source 3

...

Data Source n

Read Move

BUS SYSTEM

DMA Modul

DMA Kanal 1

DMA Kanal 2

DMA Kanal 3

DMA Kanal 4

...

DMA Kanal n

Maximale Blockgröße einer Transaction???

Anzahl an DMA-Kanälen???

Move Engine 1

Move Engine 2

Write Move

BUS SYSTEM

DMA Ziel

Data Destination 1

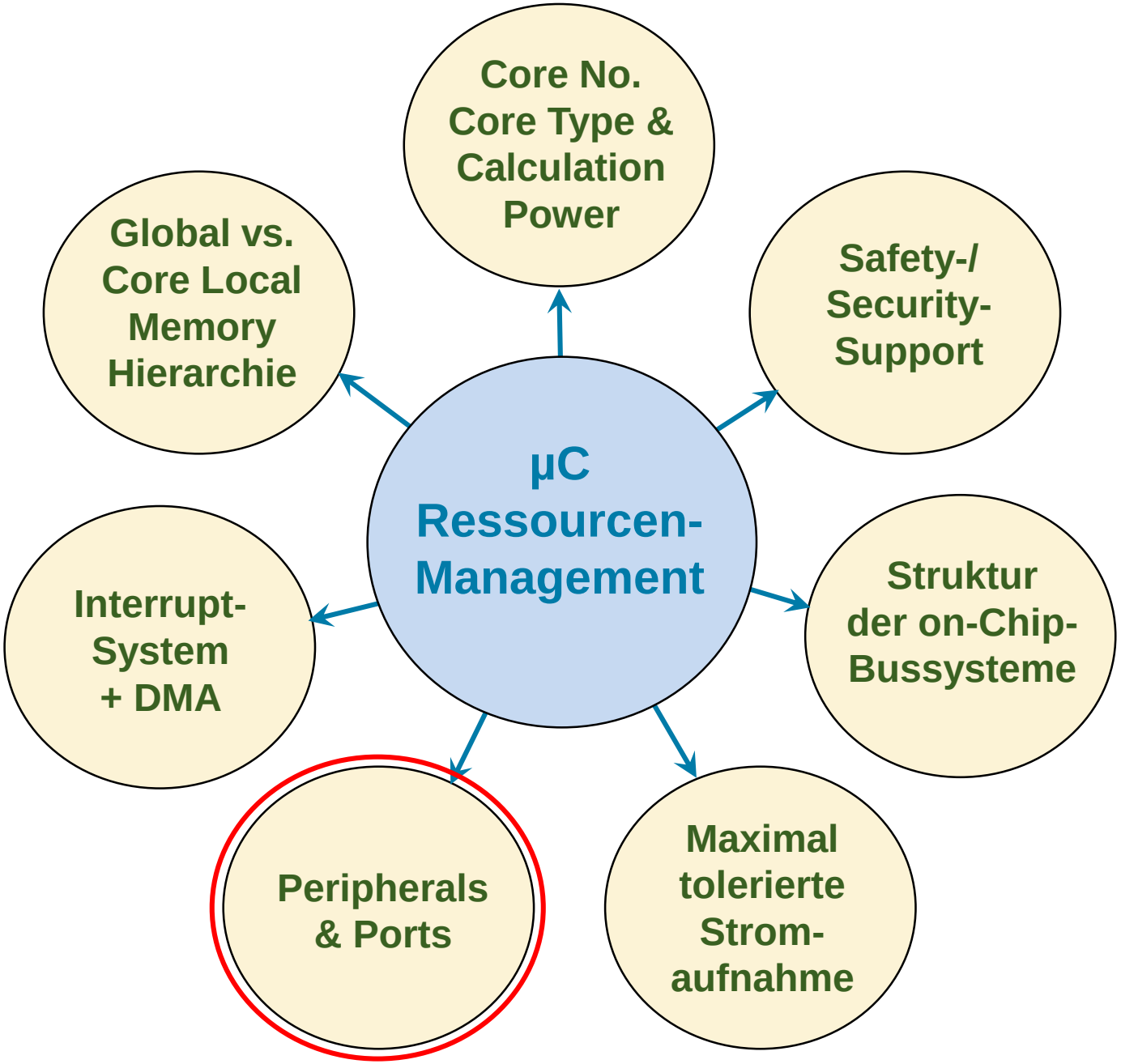
Data Destination 2

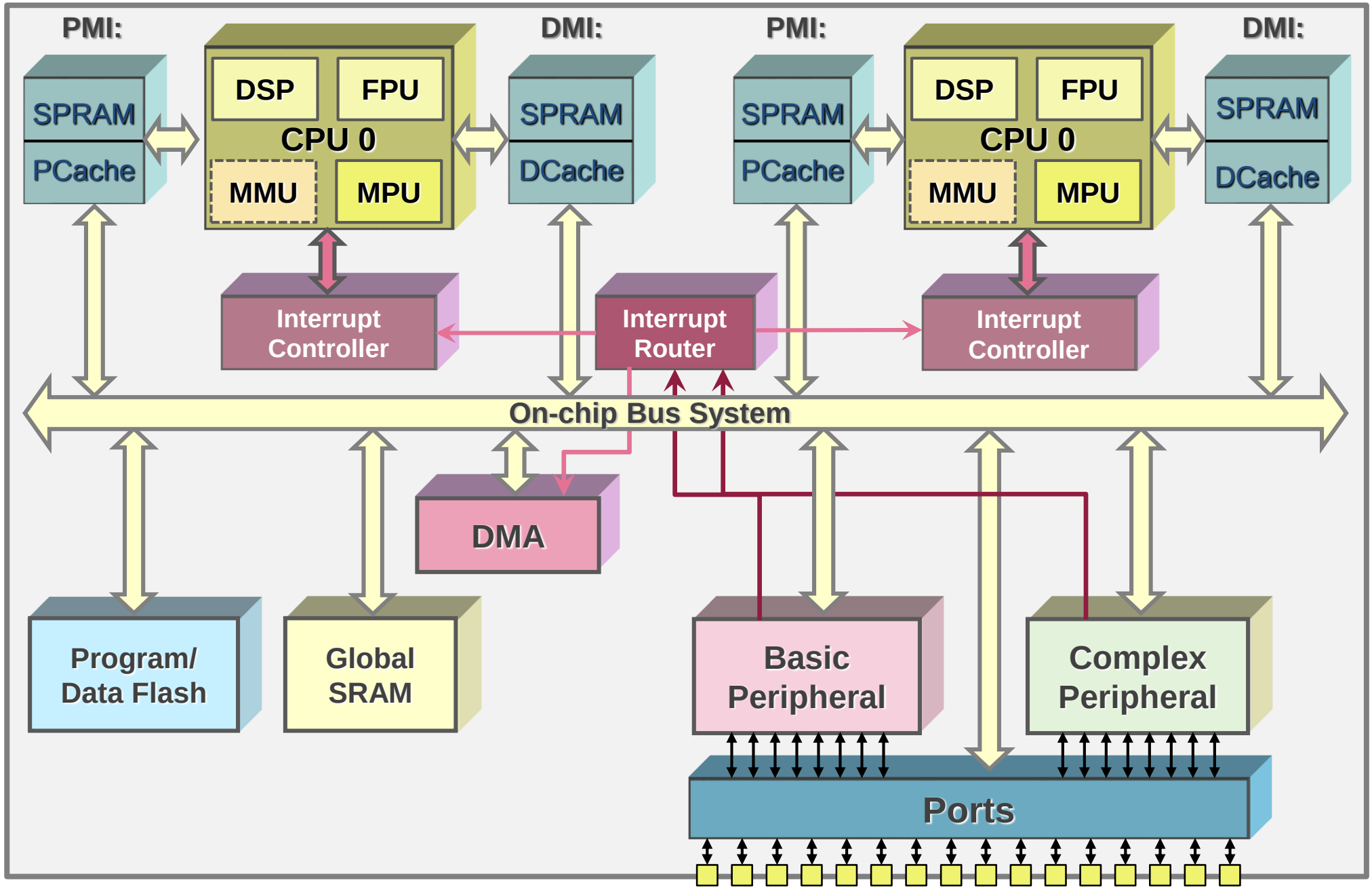
Data Destination 3

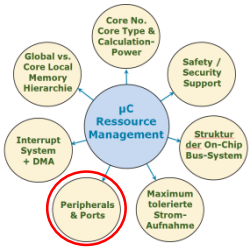
...

Data Destination n

Die 7 wichtigsten Aspekte für das Ressourcen-Management in Multicore-µC




PMI:
DMI:
PMI:
DMI:
SPRAM
PCache
DSP
FPU
CPU 0
MMU
MPU
SPRAM
DCache
SPRAM
PCache
DSP
FPU
CPU 0
MMU
MPU
SPRAM
DCache
Interrupt Controller
Interrupt Router
Interrupt Controller
On-chip Bus System
DMA
Program/ Data Flash
Global SRAM
Basic Peripheral
Complex Peripheral
Ports



Ein Peripheriemodul kann I/O-Leitungen an verschiedenen Port-Modulen haben

On-chip Bus System

Port Module A

Bus Interface

Port Register (SFR)

Port State Machine

MUX: Peri Signal Sel.

Input Decoder

Output Driver

PERI Events

Peripheral Module

Bus Interface

Peripheral Reg. (SFR)

Peripheral State Machine

MUX: Module Interface Sel.

Port Module B

Bus Interface

Port Register (SFR)

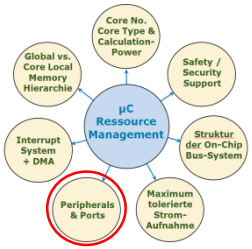
Port State Machine

MUX: Peri Signal Sel.

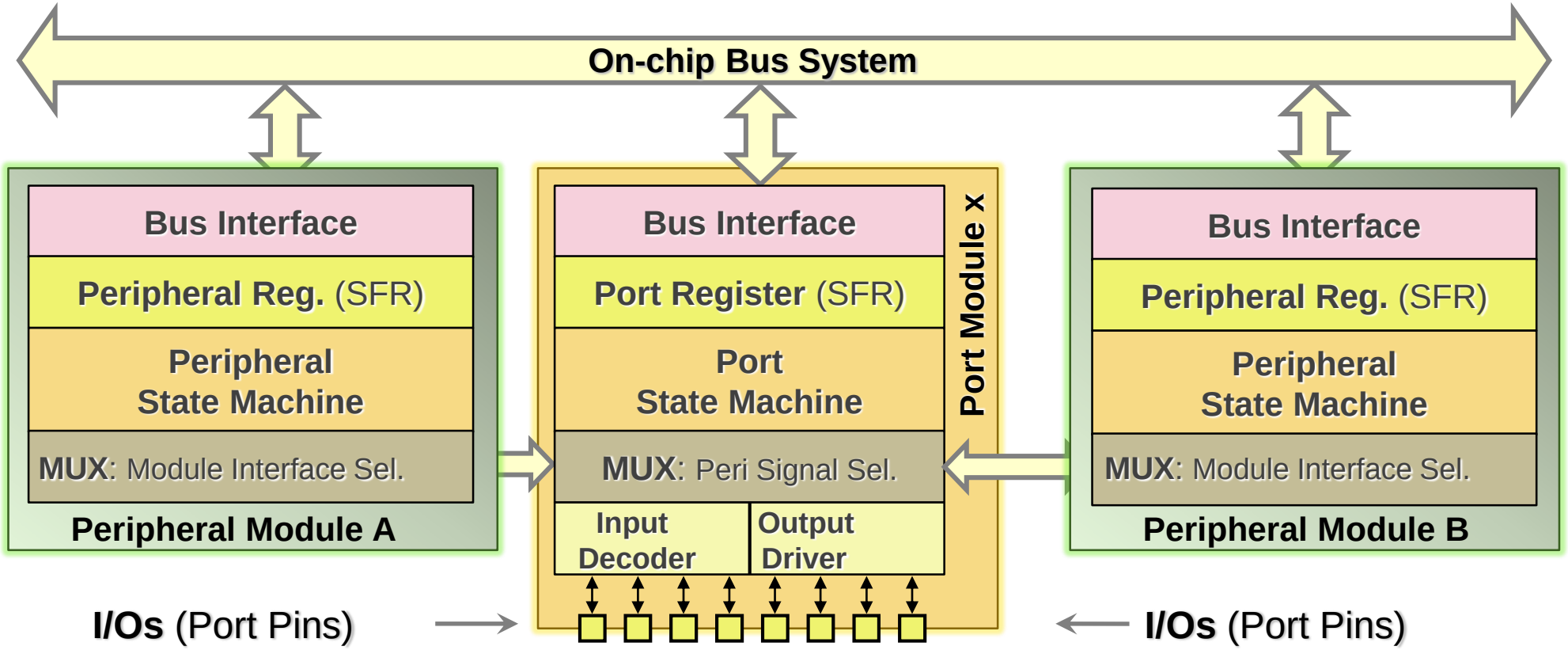
Input Decoder

Output Driver

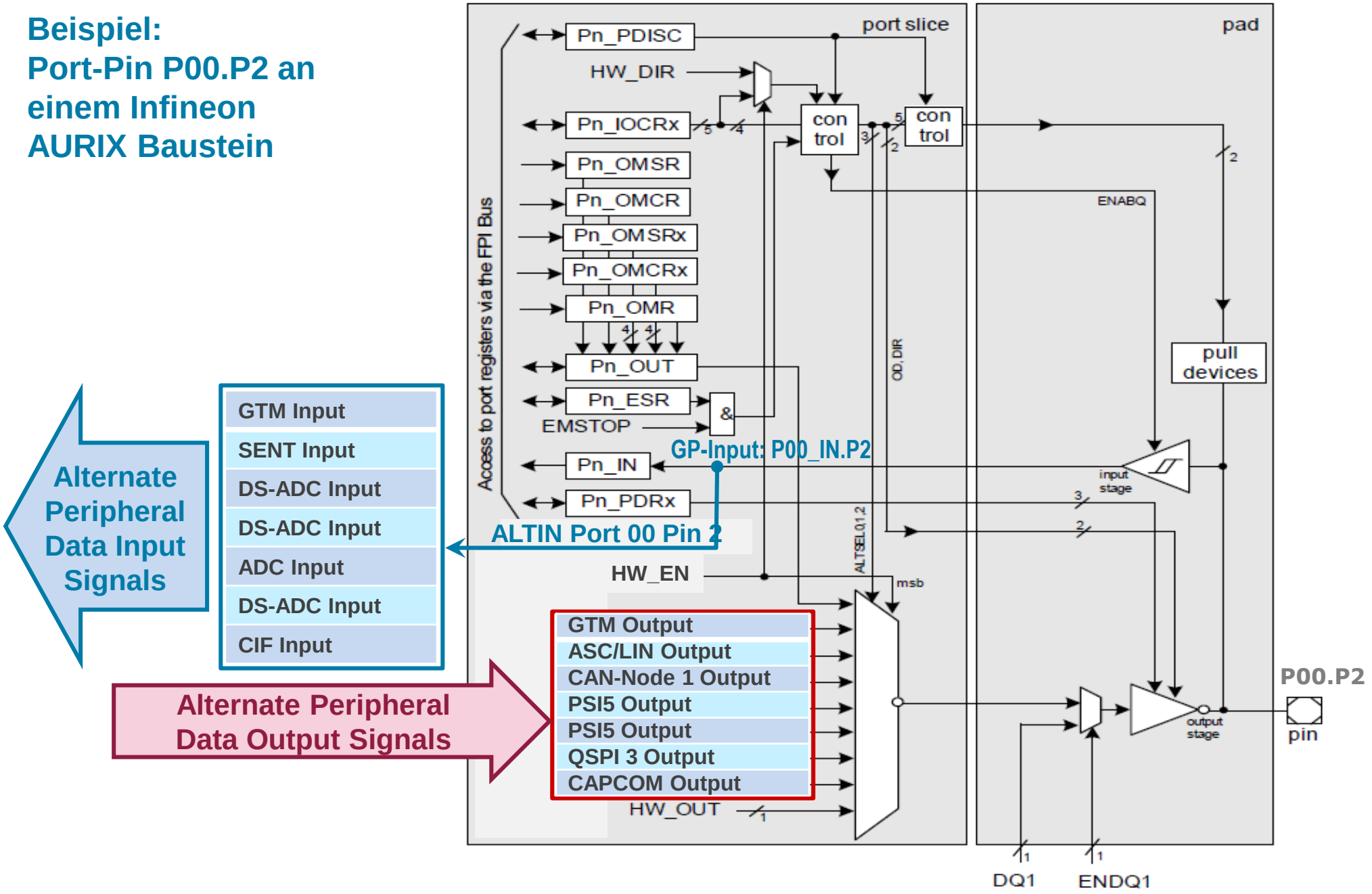
I/Os (Port Pins)

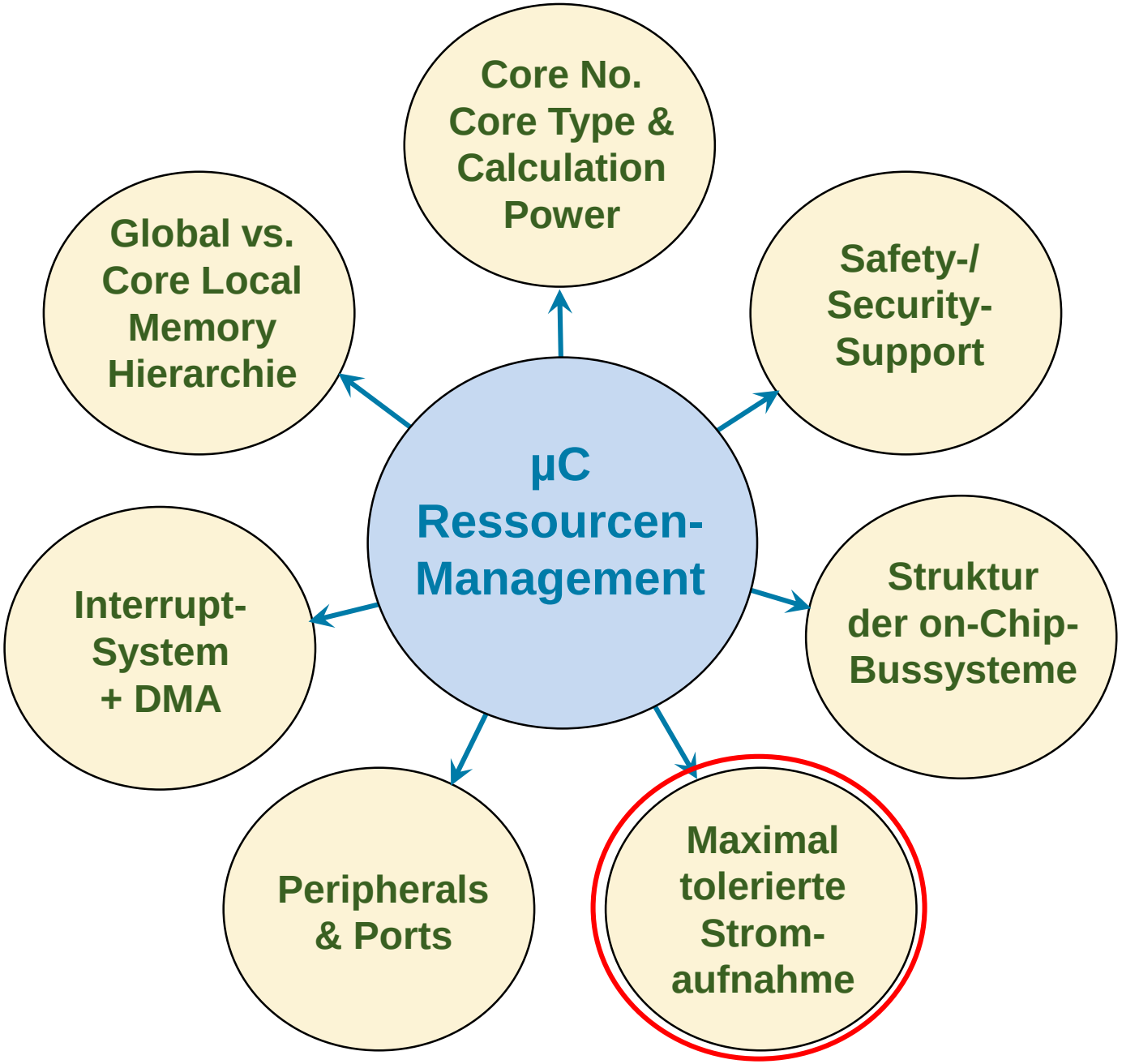


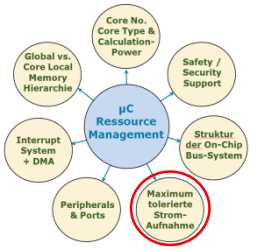
I/O-Funktionen von verschiedenen Peripherie-Modulen können an einem gemeinsamen Port-Modul verbunden sein.



Beispiel:
Port-Pin P00.P2 an einem Infineon AURIX Baustein




Core No.
Core Type &
Calculation
Power
Global vs.
Core Local
Memory
Hierarchie
Safety-/
Security-
Support
µC
Ressourcen-
Management
Interrupt-
System
+ DMA
Struktur
der on-Chip-
Bussysteme
Peripherals
& Ports
Maximal
tolerierte
Strom-
aufnahme



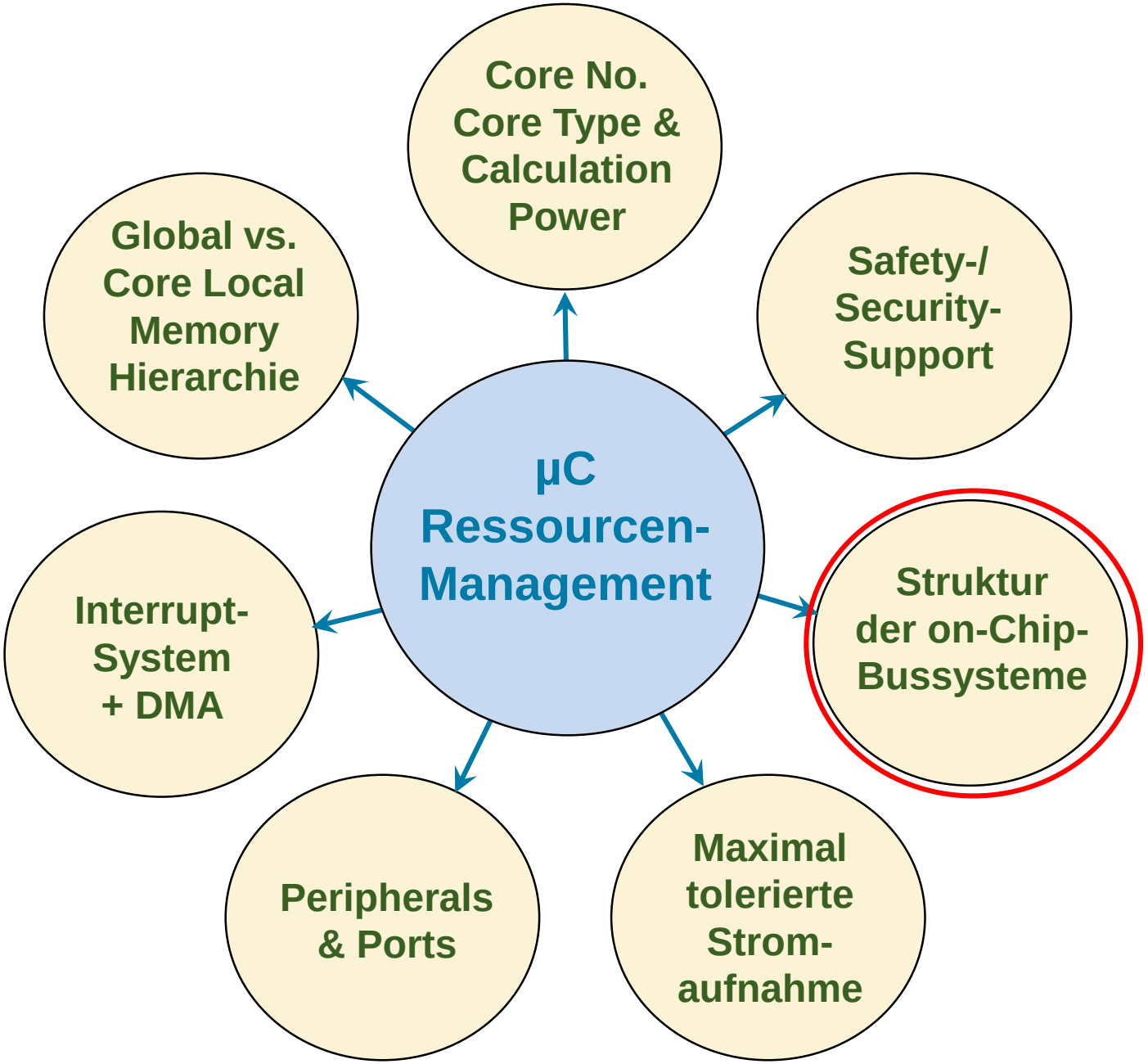
Baustein-Stromaufnahme

high

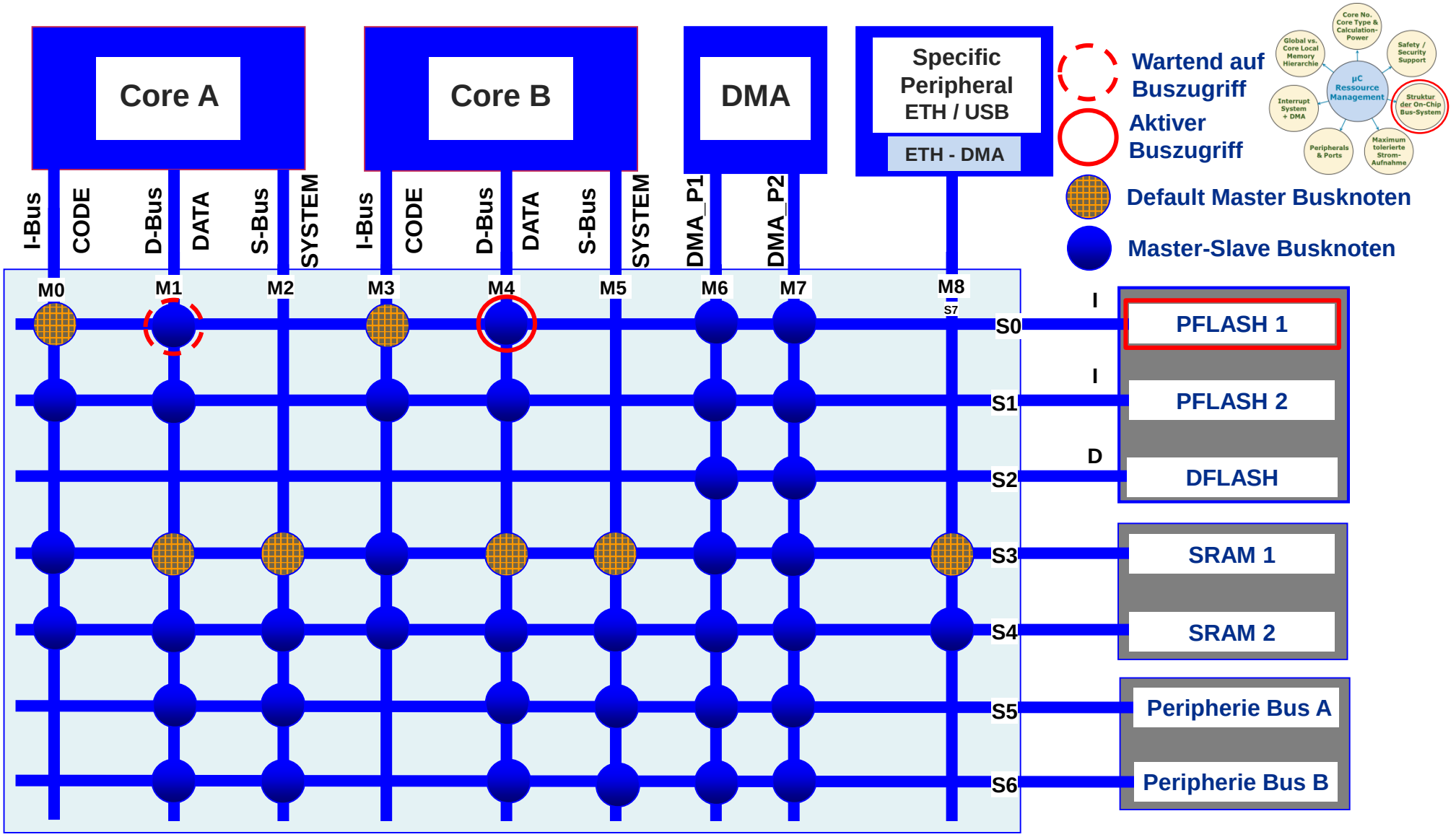
low

Hauptaspekte für die Stromaufnahme:

- Core-Taktfrequenz
- Anzahl aktiver Cores
- Peripherie-Taktfrequenz
- Anzahl aktiver Peripherie-Module
- Taktgeschwindigkeit an den Ports
- Treiberleistung an den aktiven Ausgangs-Ports
- Chip-Technologie und Chip-Design

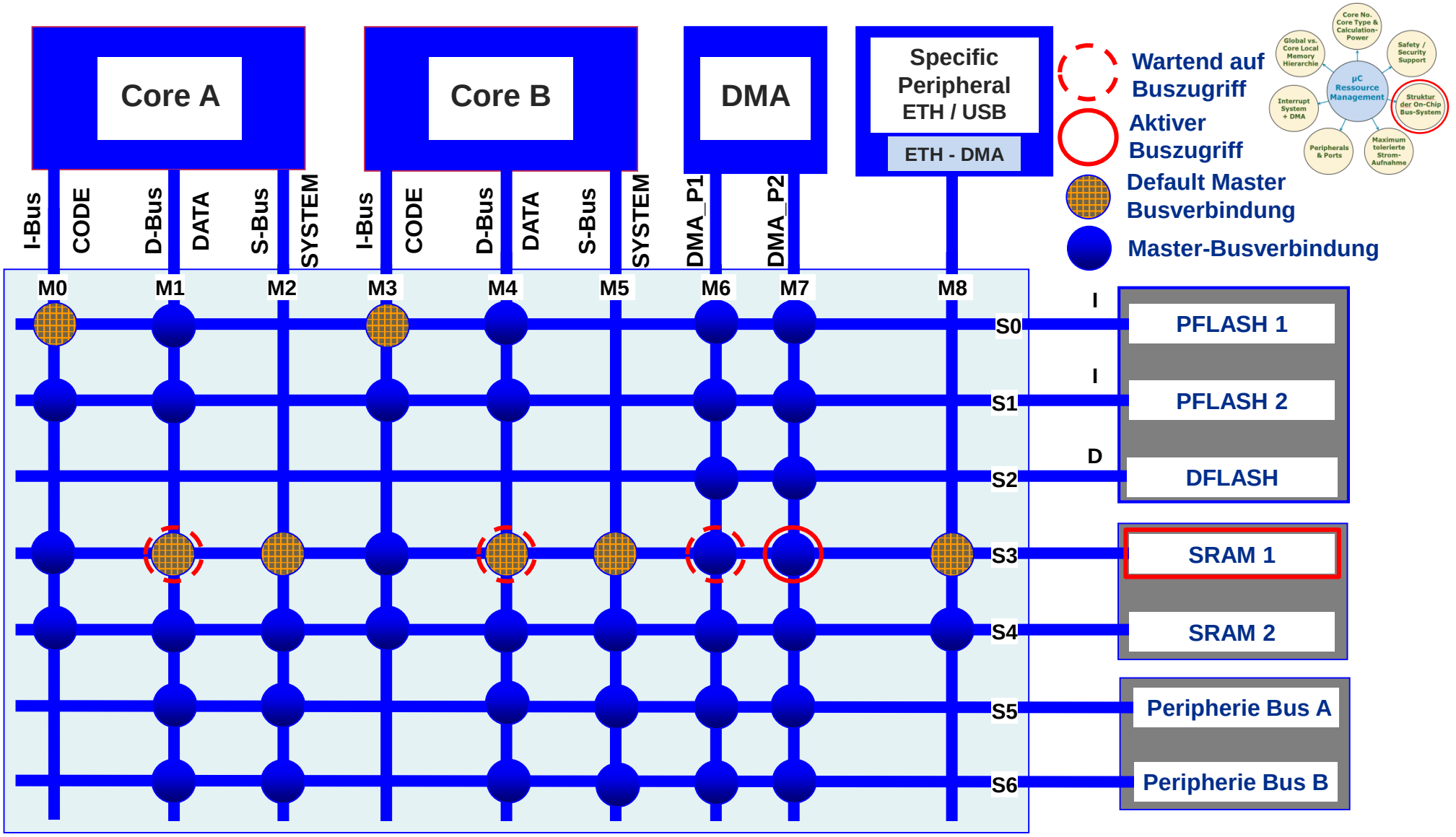

Core No.
Core Type &
Calculation
Power
Global vs.
Core Local
Memory
Hierarchie
Safety-/
Security-
Support
µC
Ressourcen-
Management
Interrupt-
System
+ DMA
Struktur
der on-Chip-
Bussysteme
Peripherals
& Ports
Maximal
tolerierte
Strom-
aufnahme

Busmatrix-Architektur – Crossbar Switch XBAR – konkurrierende RAM-Zugriffe



Wikipedia:
A crossbar switch is an assembly of individual switches between a set of inputs and a set of outputs. The switches are arranged in a matrix.

Busmatrix-Architektur – Crossbar Switch XBAR – konkurrierende Flash-Zugriff e

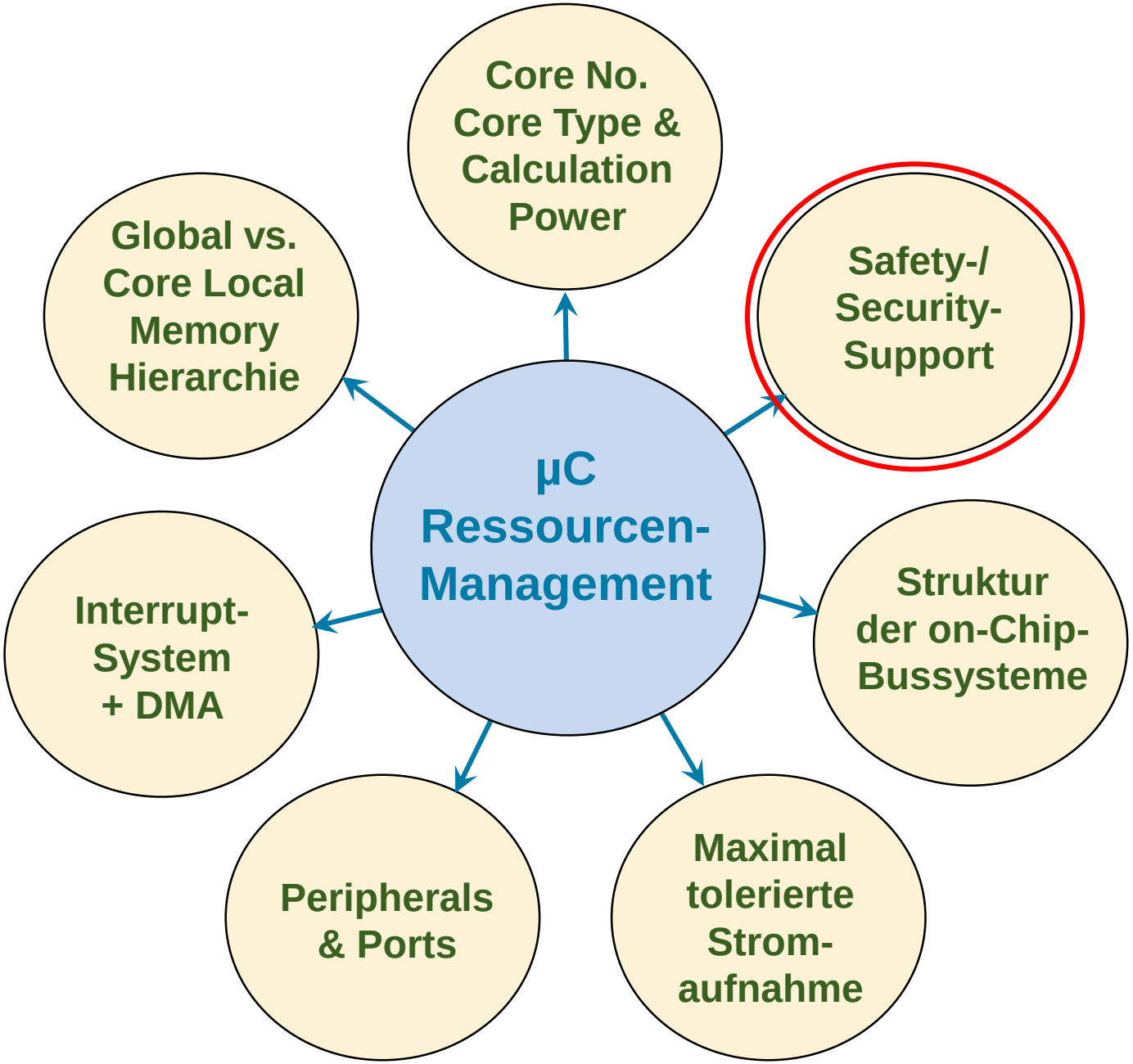


Wikipedia:
A crossbar switch is an assembly of individual switches between a set of inputs and a set of outputs. The switches are arranged in a matrix.

Analyse für das Ressourcen-Management am Bussystem:

- **Real-time Datenkommunikation:**
  - **Welche Real-time Datenkommunikation kann auftreten?**
  - **Wer generiert die Daten?**
  - **Wo sollen die Daten gespeichert werden?**

- **Bus-Master-Kommunikation:**
  - **Welcher Bus-Master kommuniziert welche Datenmengen über welchen Datenbus?**
  - **Wir häufig werden die Datenmengen kommuniziert?**


Core No.
Core Type &
Calculation
Power
Global vs.
Core Local
Memory
Hierarchie
Safety-/
Security-
Support
µC
Ressourcen-
Management
Interrupt-
System
+ DMA
Struktur
der on-Chip-
Bussysteme
Peripherals
& Ports
Maximal
tolerierte
Strom-
aufnahme

Kommen Anforderungen zu **Safety** und **Security** ins Spiel, ist eine umfassende Anforderungsanalyse für diese Aspekte sehr wichtig.

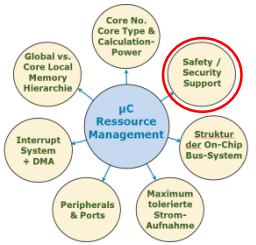
Zum Erreichen des **ASIL-Levels C** bestehen besondere Safety-Anforderungen.
Die Verarbeitung einer Safety-relevanten Software-Applikation und damit die Generierung von Ergebnissen in **einem Safety-Core** müssen mit den Ergebnissen der zeitversetzten Programmbearbeitung in einem **Checker-Core** verglichen werden.

Dies ist notwendig, damit eine eventuell auftretende Störung während der Programmverarbeitung im Main-Core erkannt werden kann.

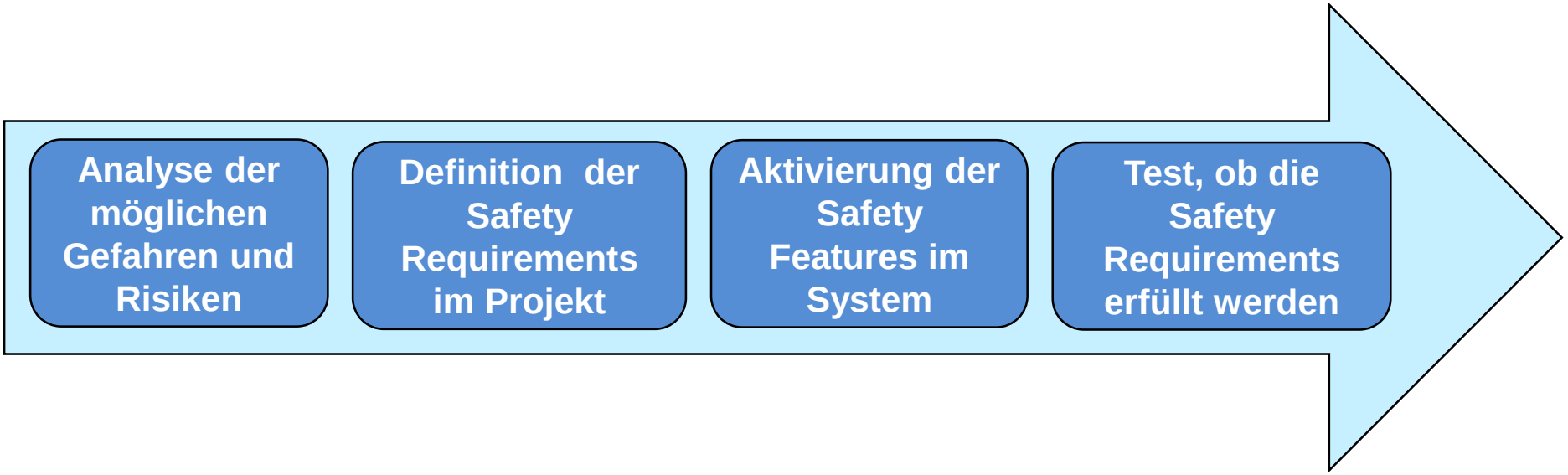
Bei einer Fehlererkennung muss das System eine geeignete Fehlerantwort generieren.

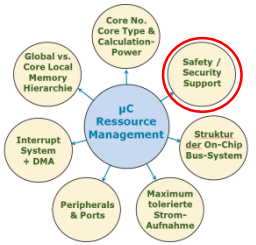
Diese **Fehlerbehandlung** muss in speziellen Fällen das **System,** ohne die Verarbeitung von Software, in einen **sicheren Zustand** versetzen können.

Hier wird eine **Safety-Hardware** (Safety Management Unit) benötigt, die für jeden erkannten Fehler automatisch eine **vorgewählte/definierte Fehlerreaktion** auslöst.



Projekt-Entwicklung für Safety-relevante Systeme





Safety-Anforderung (Safety-Standard)

Benötigter Safety-Support

Safety-Standard Beispiele:

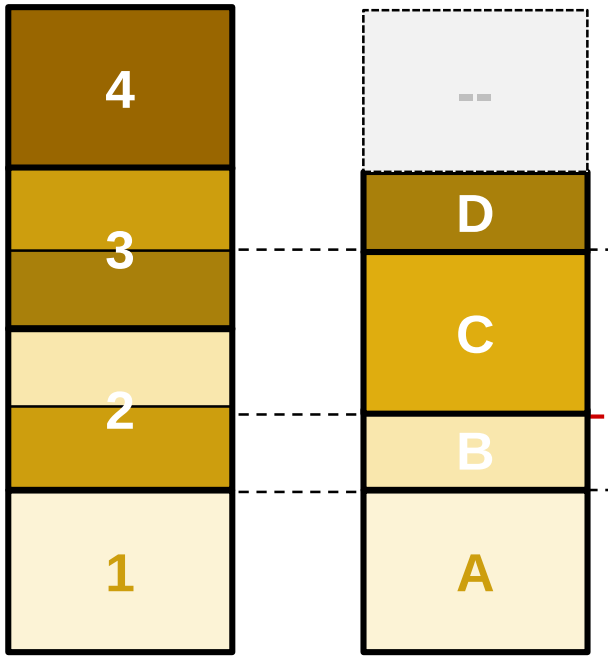
Einige Beispiele für Safety-Anforderungen:

Safety Level High

Low

SIL
IEC 61508

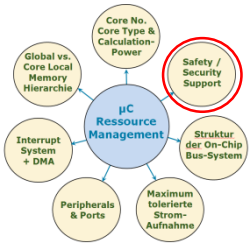
ASIL
ISO 26262



Zusätzliche Anforderungen für ASIL-C:
- Checker Core, Flash ECC Correction
- MPU: Peripheral Memory Protection
- I/O Monitoring
- Clock- und Spannungs-Monitoring
- Redundant Peripherals

Anforderungen für ASIL-B:
- MPU: Core-Memory Protection
- Safety Error Detection und Monitor Units in µC SRAM & Flash (ECC)
- SW Self-Tests

MICRO CONSULT



Projekt-Security: Sicherheitssysteme für Steuerungen

Security-Anforderung

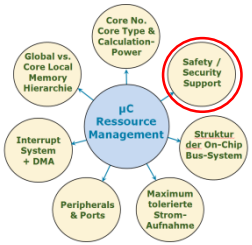
Systemschutz vor
- nicht autorisierten Manipulationen des Systems und dessen Daten
- Modifikationen der Safety-Applikation

Zugriffsschutz resp. Schutz
- der Privatsphäre der Applikation bzw. Treiberfunktionen
- des geistigen Eigentums von Herstellern und Zulieferer

Schutz der Kommunikation:
- On-board
- Extern über Bussysteme

Vermeidung von System-Blockaden
durch Interferenzen von Security-System und Applikations-Ressourcen

Benötigter Security-Support



Speicher-Zugriffsschutz

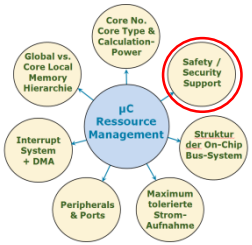
Zugriffsschutz für Peripherie-Module

Passwortgeschützte Boot Modes

Asymmetrischer/symmetrischer Kryptographie-Prozessor

Random Number Generator RNG

Verwendung eigener Ressourcen für das Protection System bzw. High Security Modul HSM: Separate Safety-CPU incl. Speicher



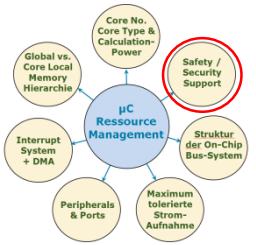
Was soll bzw. muss geschützt werden?

Welche Angriffs-Potentiale gibt es?

Was sind die Security-Ziele?

Welche Bedrohungen gibt es?

Was sind mögliche Angriffspunkte?



Applikation mit verschiedenen ASIL-Anforderungen

ECU 1
ASIL C Applikation

ECU 2
ASIL B Applikation

ECU 3
ASIL A Applikation

SW-Implementierung in einer Multicore-Architektur

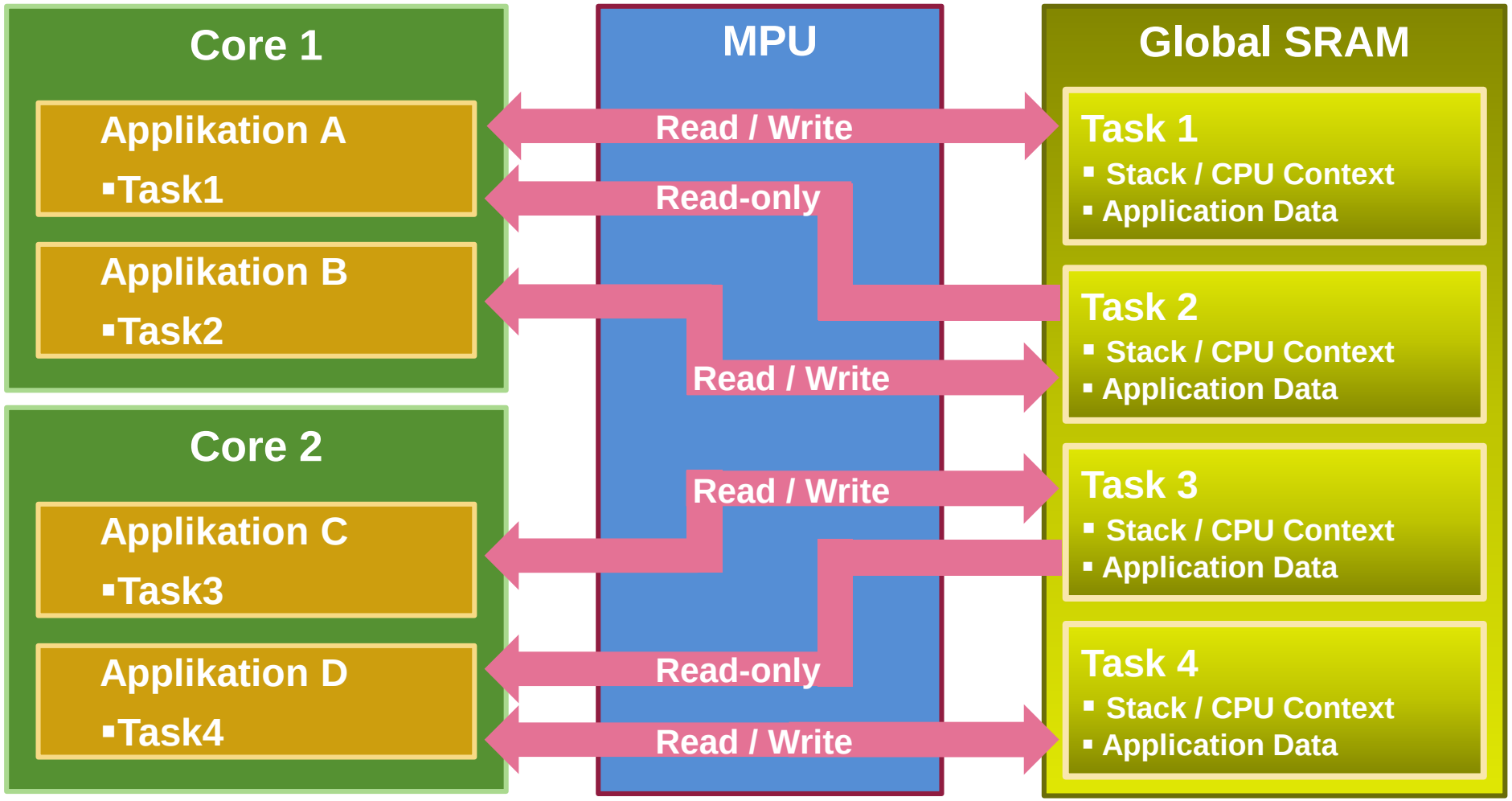
Checker Core

Core 1
ASIL C Applikation

Core 2
ASIL B Applikation

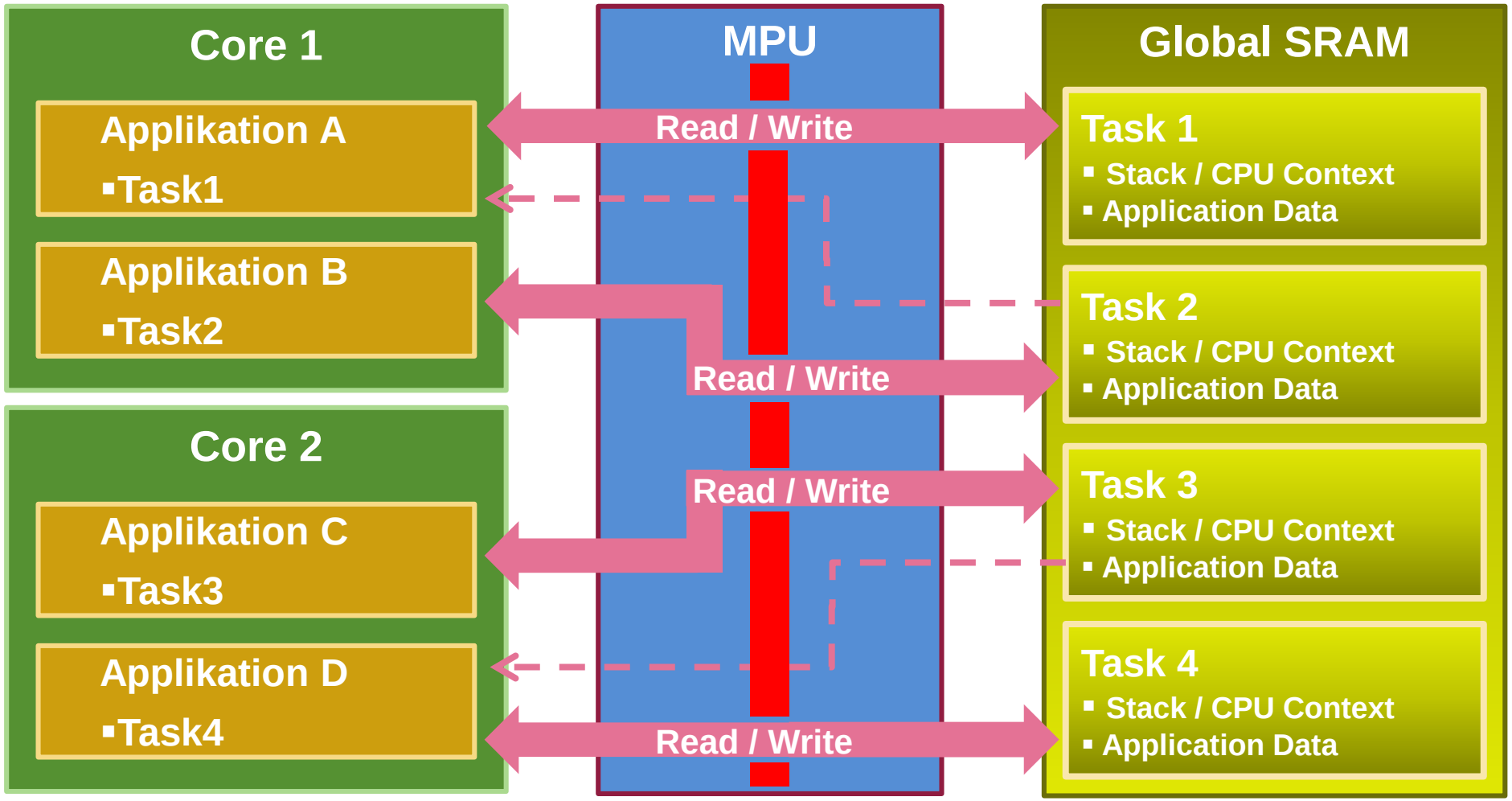
Core 3
ASIL A Applikation

Multicore-µController



Ressourcen-Management für MPUs:

- Ist die **MPU-Implementierung Core-lokal** oder **global** im **Baustein?**
- Über wie viele **Memory Protection Sets verfügt die MPU?**
- Wie viele **Memory-Bereiche** kontrolliert ein **MPU-Set?**



Ressourcen-Management für MPUs:

- Ist die **MPU-Implementierung Core-lokal** oder **global** im **Baustein?**
- Über wie viele **Memory Protection Sets verfügt die MPU?**
- Wie viele **Memory-Bereiche** kontrolliert ein **MPU-Set?**

Die **erfolgreiche Mikrocontroller-Auswahl** basiert auf einer…

Festlegung der **Entwicklungsziele /Qualitätsvorgaben**:

Skalierbarkeit des neuen **Architektur-Designs**

+

Wiederverwendbarkeit der **Software**

Umfangreiche Anforderungsanalyse für das **Projekt**

Priorisierung aller **Anforderungen**

Auswahl von **kompetenten Partnern**

Coaching: Multicore-Mikrocontroller-Auswahl

Maßgeschneiderter Support für Ihre Mikrocontroller-Auswahl:

1. Wir nehmen Ihre Anforderungen auf.
2. Wir werten Ihre Anforderungen aus.
3. Wir suchen für Sie die Architektur aus, die Ihre Anforderungen am besten erfüllt.
4. Wir stellen die Ergebnisse der getroffenen Auswahl vor.
5. Wir zeigen Ihnen die zu bewältigenden Herausforderungen auf.
6. Wir bieten Ihnen an Ihre Anforderungen angepasste Trainings an.
7. Wir begleiten Sie auf Wunsch beratend bei Ihrer Migration.

Mehr Information zum Coaching

Technisches Training – Experience Embedded:

Wir bieten Ihnen ein großes Themenspektrum an offenen Trainings an:

1. **Anforderungsanalyse**
2. **Software-Architektur**
3. **Multicore-Mikrocontroller-Grundlagen**
4. **Safety - Funktionale Sicherheit**
5. **Betriebssystem-Grundlagen**
6. **Embedded C** und **Embedded C++**

www.microconsult.de