Untersuchungen zu In-Circuit Debug-Möglichkeiten für die Intel® XScale™ Mikroarchitektur

Kai Schicktanz

Kai.Schicktanz@mailbox.tu-dresden.de

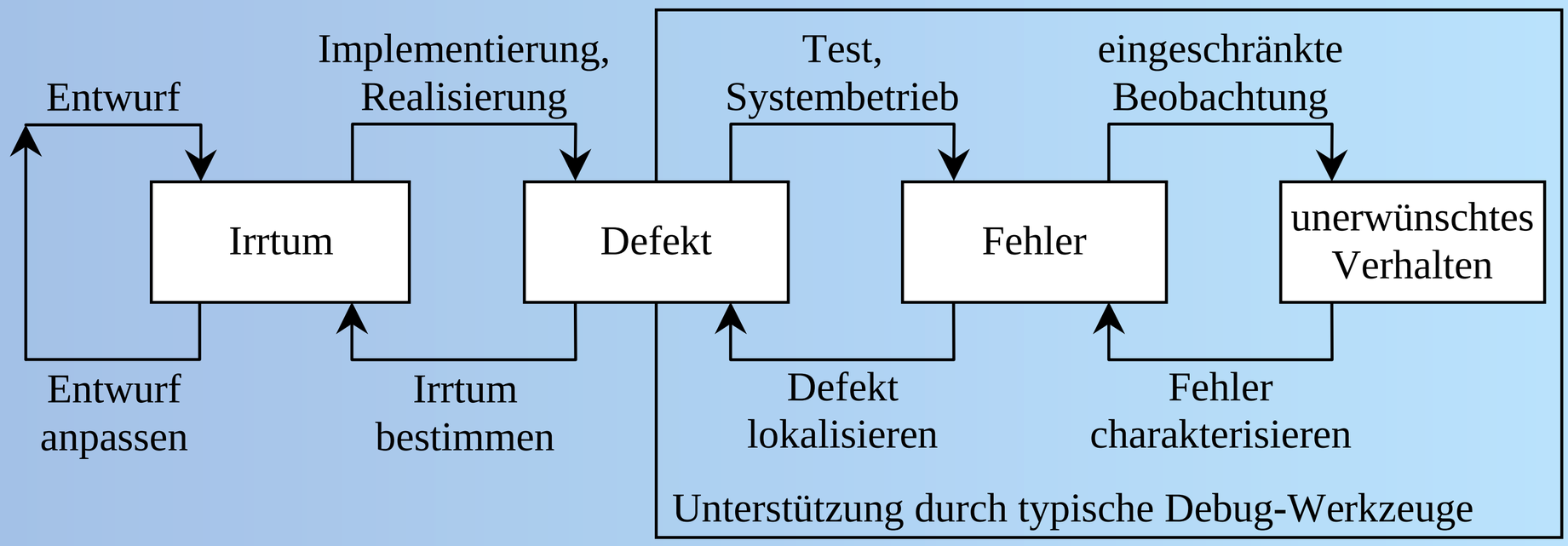
Gliederung

- Motivation / Zielstellung
- Debugging / In-Circuit Debugging
- Testschnittstelle nach IEEE 1149.1
- Intel XScale Mikroarchitektur / Debug-Konzept
- Softwareentwurf / Implementation
- Leistungsbewertung
- Zusammenfassung / Ausblick

Motivation / Zielstellung

- Entwicklungstrend:
  - Software für eingebettete Systeme wird zunehmend komplexer
  - Bei gleich bleibender Fehlerrate steigt dadurch die Gesamtfehlerzahl
- Typische Gegenmaßnahmen:
  - Prozess- und Qualitätsmodelle im gesamten Entwicklungsprozess
  - Formale Methoden in Entwurfs- und frühen Entwicklungsphasen
  - **Debugging**, Simulation, und statische Analyse in der Testphase
  - Fehlertoleranzmaßnahmen in der Betriebsphase
- Generische Debug-Werkzeuge für eingebettete Software:
  - Abstraktion typischer Operationsprinzipien und Architekturmerkmale
  - Generische Werkzeuge sind realisierbar, Anpassungen an die spezifischen Merkmale einer Prozessorarchitektur zumeist nötig
- Zielstellung:
  - Untersuchungen zum Debugging der Intel XScale Architektur
  - Prototypische Erweiterung des vorhandenen Debug-Werkzeuges „Universal Debug Engine“ (UDE) am Beispiel des Intel PXA 255

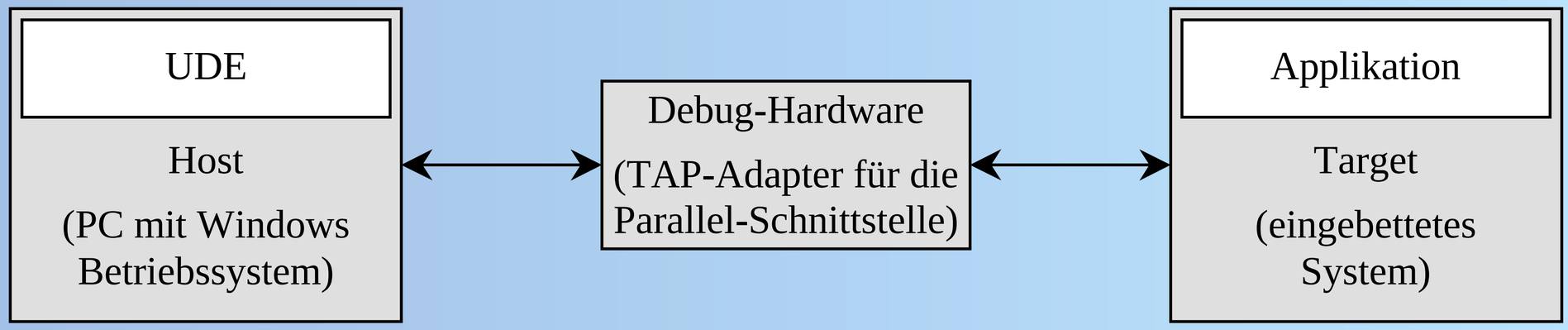
- Modellvorstellung zum dynamischen Debugging:



- Wesentliche Aufgaben eines Debug-Werkzeuges:
  - Beobachtbarkeit im Zeit- und Zustandsbereich sichern
  - Effiziente Test- und Stimulationsmöglichkeiten bereitstellen
- Zentrale Forderungen an ein Debug-Werkzeug:
  - Hohe Verlässlichkeit und Qualität bei großem Funktionsumfang
  - Unerwünschte Beeinflussung des untersuchten Systems minimieren

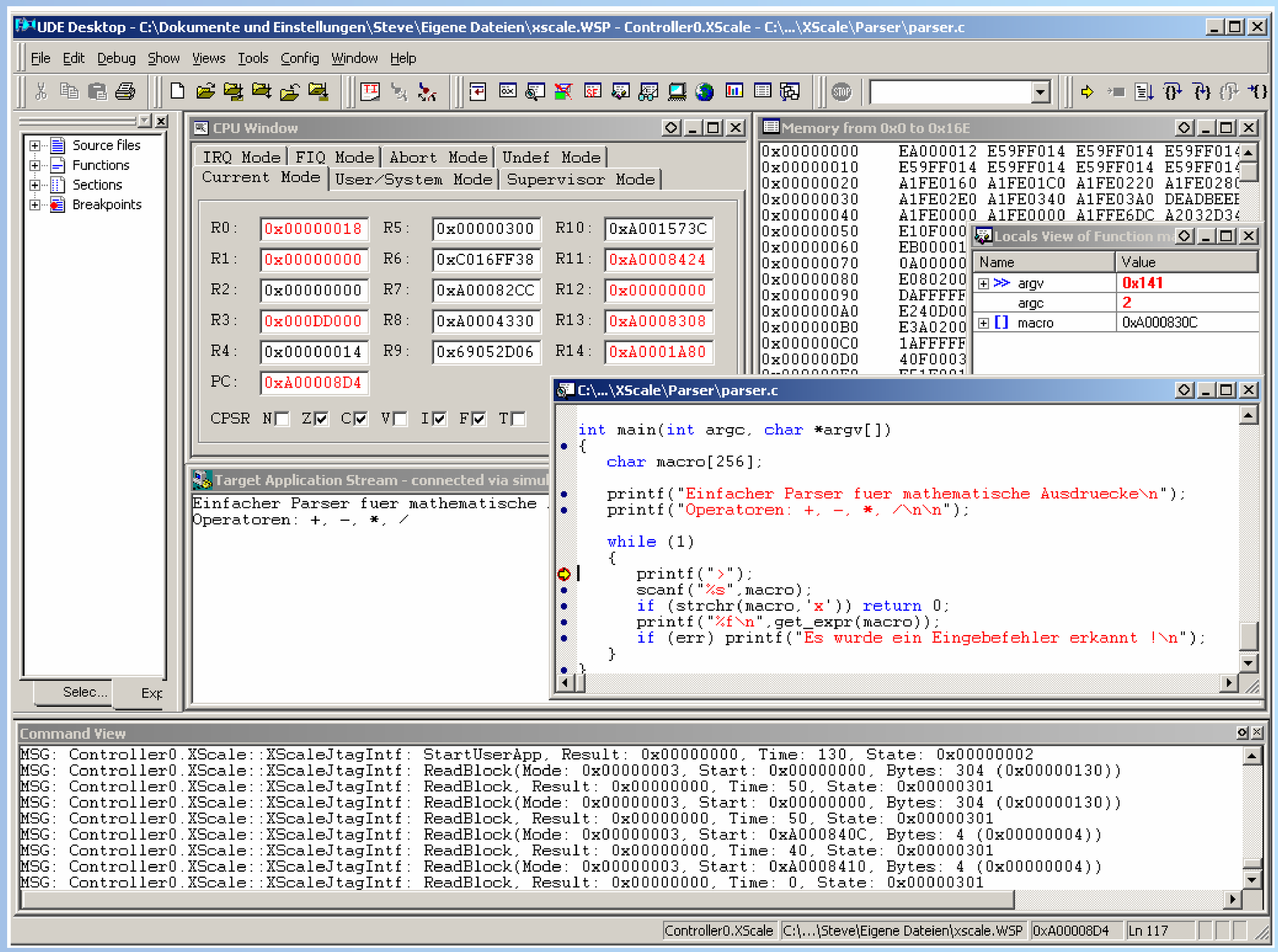
In-Circuit Debugging

- In-Circuit Debugging:
  - Debugging von Software direkt auf der eingebetteten Hardware



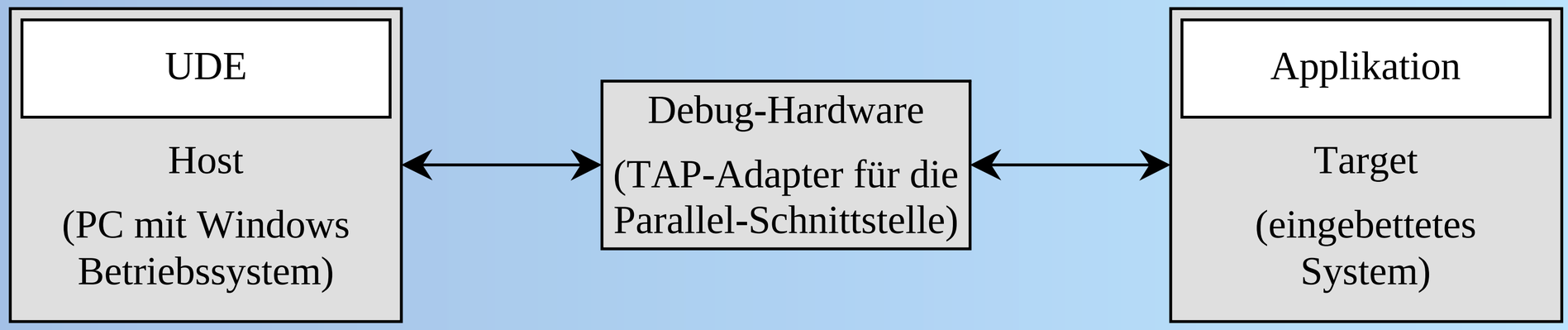
- Typische Aufteilung des Gesamtsystems:
  - Eingebettetes System (Target):
    - Ausführung der debuggten Applikation unter minimaler Beeinflussung
    - Möglichkeiten der Analyse und Manipulation: Debugger-Monitor, On-Chip Debug-System, ...
  - Debug-Hardware
    - komplexe Systeme typisch, hier nur elementare Hardware verwendet
    - Kommunikation über spezielle Schnittstellen (IEEE 1149.1, IEEE Nexus)
  - Host:
    - Ressourcen für aufwändige Teilprozesse und Werkzeug-Komponenten

Universal Debug Engine



In-Circuit Debugging

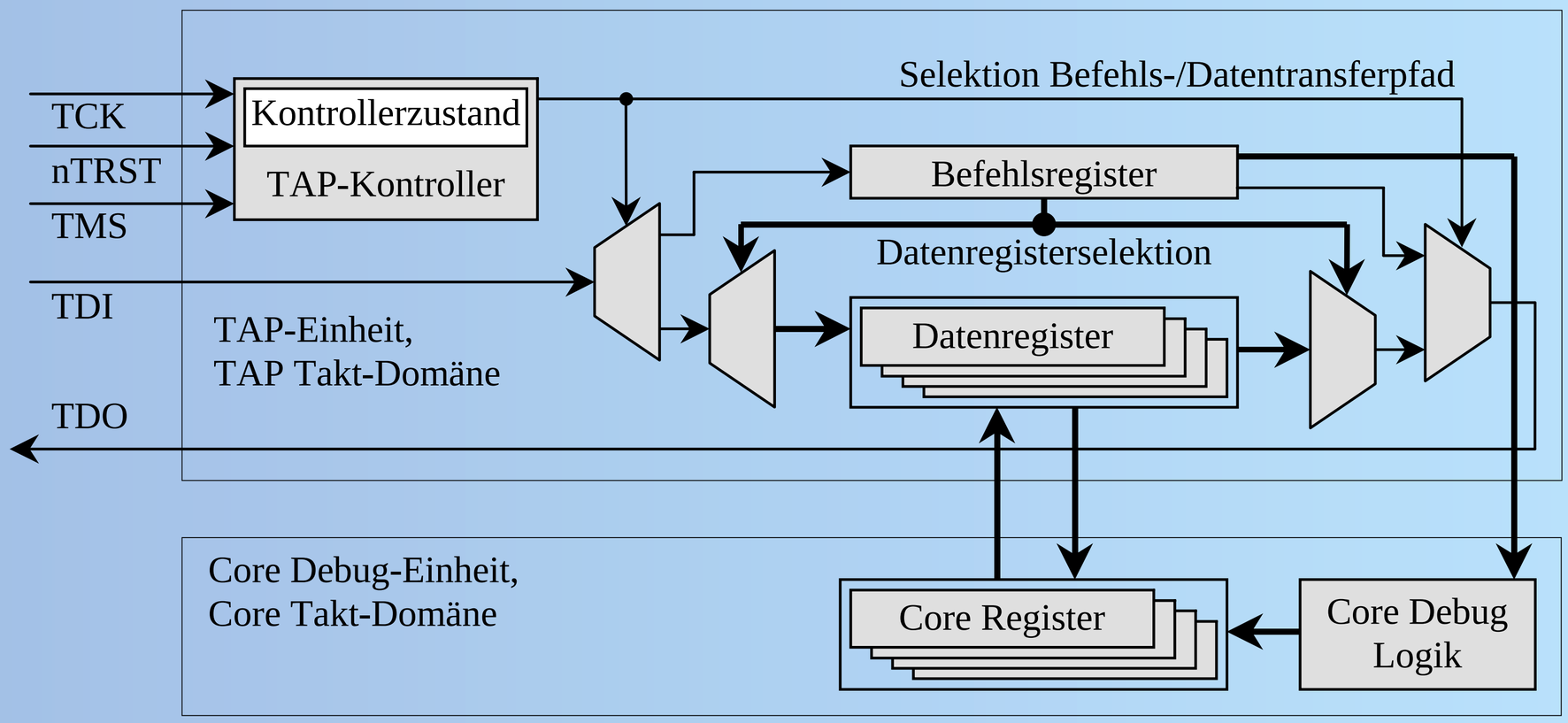
- In-Circuit Debugging:
  - Debugging von Software direkt auf der eingebetteten Hardware



- Typische Aufteilung des Gesamtsystems:
  - Eingebettetes System (Target):
    - Ausführung der debuggten Applikation unter minimaler Beeinflussung
    - Möglichkeiten der Analyse und Manipulation: Debugger-Monitor, On-Chip Debug-System, ...
  - Debug-Hardware:
    - komplexe Systeme typisch, hier nur elementare Hardware verwendet
    - Kommunikation über spezielle Schnittstellen (IEEE 1149.1, IEEE Nexus)
  - Host:
    - Ressourcen für aufwändige Teilprozesse und Werkzeug-Komponenten

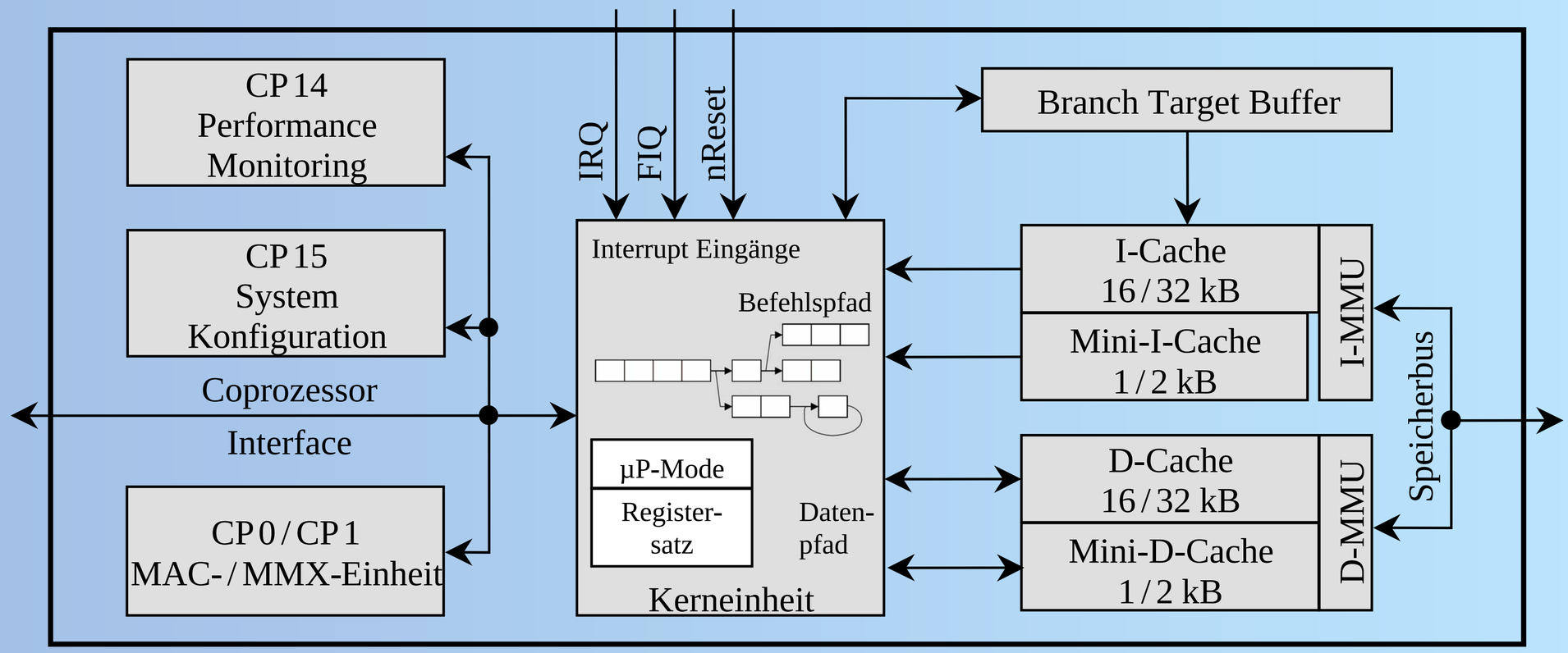
Testschnittstelle nach IEEE 1149.1

- Joint Test Action Group (JTAG):
  - Definition eines On-Chip Test Access Port (TAP)
  - Bestandteile: TAP-Kontroller, Instruktions- und Datenregister
  - Externe Signale: TCK, TMS, TDI, TDO, nTRST (optional)
  - TAP-Befehle: ID-Code, Sample, ..., spezifische Befehle (Debugging)

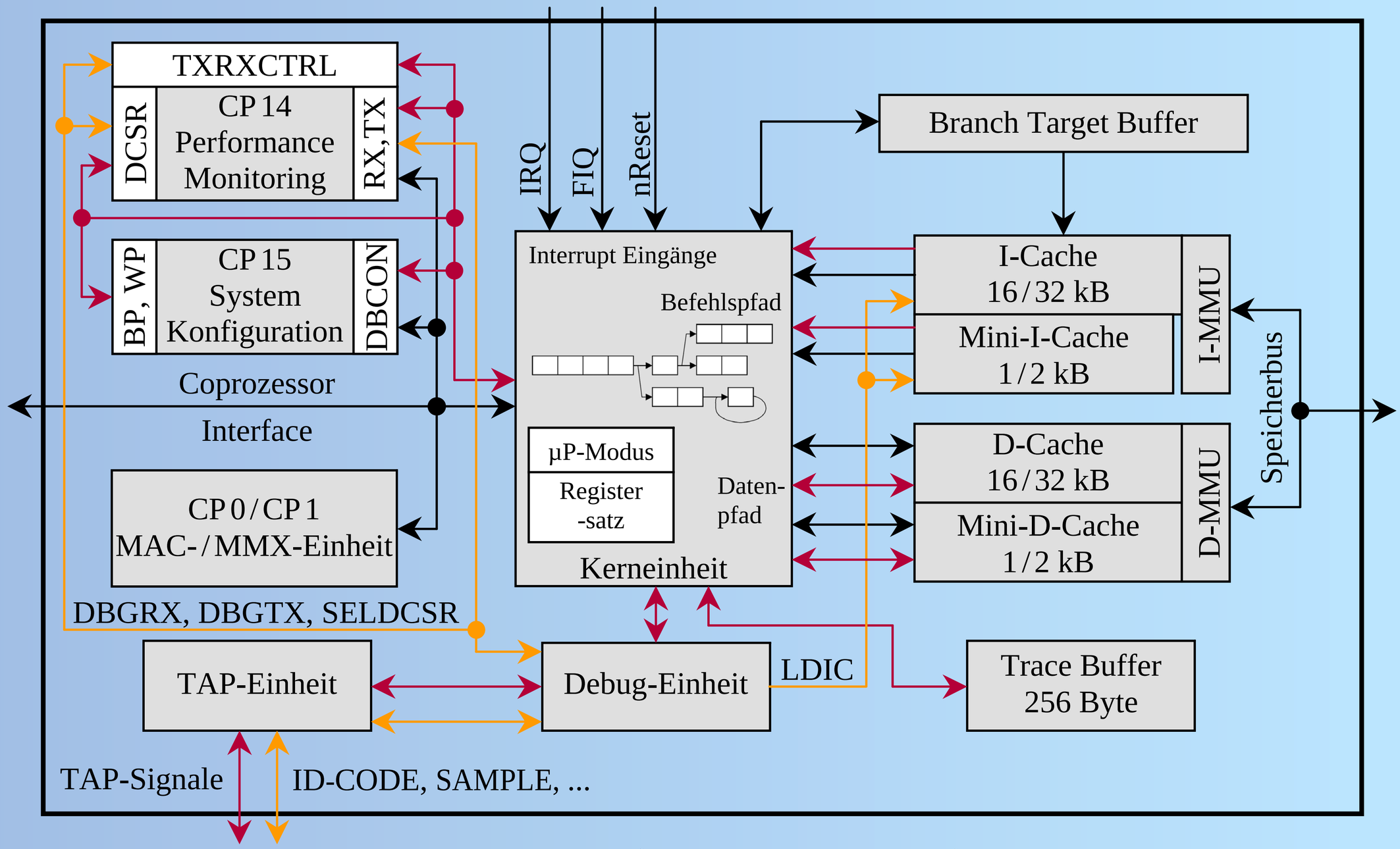


Intel XScale Architektur

- Intel XScale Architektur:
  - Mikroprozessorarchitektur, basierend auf der ARM Architektur (5TE)
  - Baustein applikationsspezifischer Standardprodukte (ASSP) für Embedded-, Mobil- und Netzwerkanwendungen
- Wesentliche Prozessorkomponenten:



On-Chip Debug-System der Intel XScale Architektur



Implementationsanforderungen und -umfang

- Anforderungen:
  - Umzusetzender Funktionsumfang vergleichbar mit bereits vorhandenen Lösungen (ARM V4), jedoch prototypischer Charakter
  - Ausschließliche Verwendung von exklusiv für das Debugging vorgesehener Hardware, Minimierung der Beeinflussung des Targets
  - Kompatibilitäts-, Flexibilitäts- und Leistungsanforderungen
- Umfang der Implementation:
  - Core-Komponente (Architekturserver):
    - Assembler, Disassembler, Komponente zur Assembler-Einzelschrittausführung
    - Erweiterung des Befehlssatzes gegenüber der unterstützten ARM-Architektur
    - Geringe Bedeutung, vorhandene Core-Komponente wird weiterhin verwendet
  - Targetinterface:
    - Prozessorspezifische Komponente zum Zugriff auf das Target
    - Debug-Konzepte von ARM und XScale zeigen grundlegende Unterschiede
    - Neuimplementation, Erhaltung struktureller Eigenschaften und Schnittstellen
  - Debugger-Monitor (Debug-Handler):
    - Neuimplementation unter Beachtung zahlreicher Restriktionen der Hardware

Entwurfsdiskussion / Konzept der Co-Routine

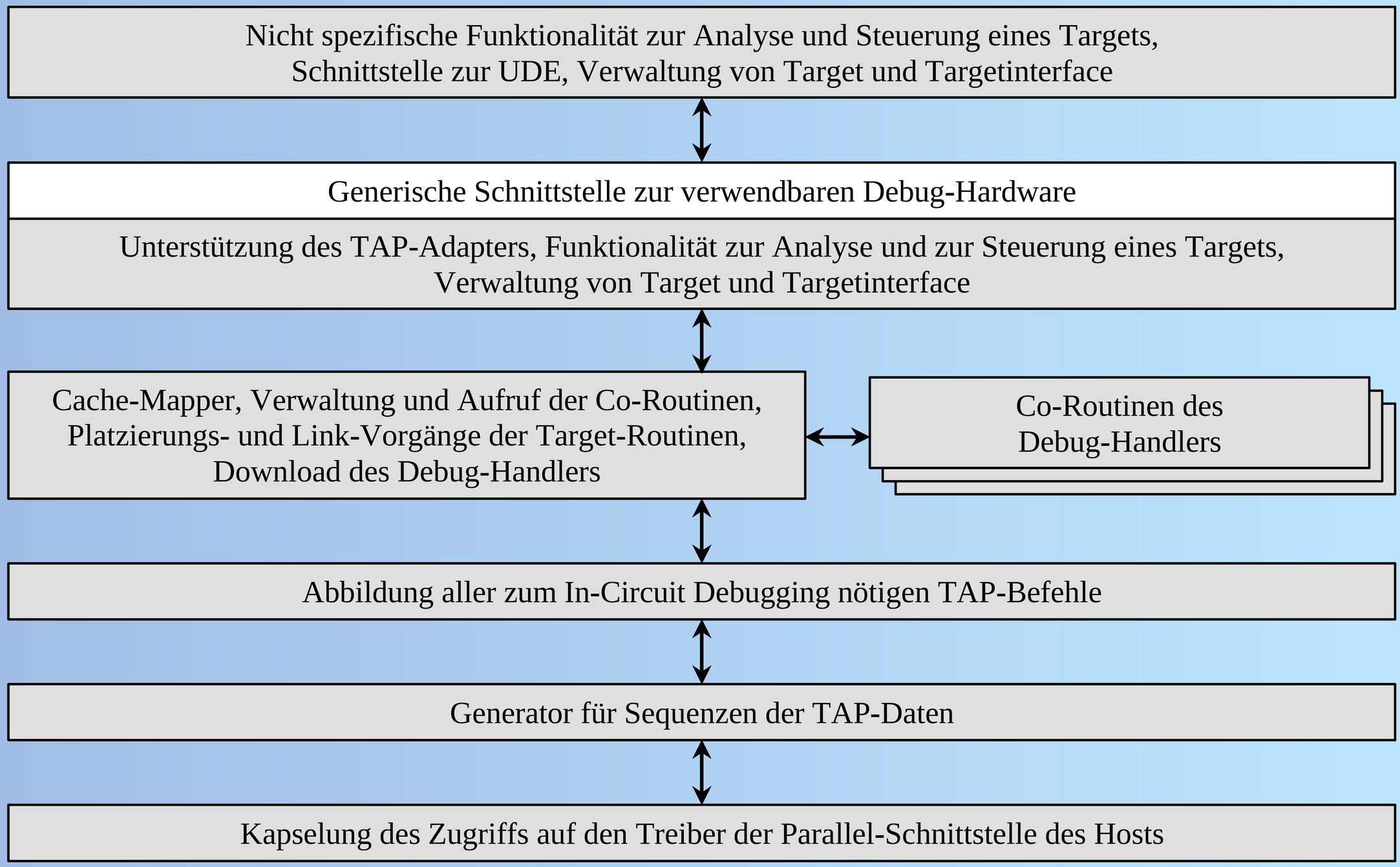
- Zentrale Entwurfsentscheidung:
  - Statische Realisierung:
    - Monolithischer Debug-Handler, Installation während des Prozessor-Resets
    - Einfache Lösung, minimaler Aufwand, Cache lässt sich optimal ausnutzen
    - Wenig flexibel (Funktionserweiterung), Größenbeschränkung des Mini-I-Caches
  - Dynamische Realisierung:
    - Partitionierung des Debug-Handlers in Routinen, Installation nach Bedarf
    - Hohe Flexibilität, einfache Erweiterbarkeit, kaum Größenbeschränkung
    - Aufwändige Lösung, Cache kaum optimal nutzbar, zusätzlicher Funktionsumfang
- Co-Routine:
  - Direkte Zuordnung einer Target- zu einer Host-Routine
  - Strukturierung der Softwarekomponenten auf Host und Target
  - Partitionierung des Kommunikationsprotokolls
  - Attribute: - fixiert, statisch, dynamisch
    - selbstplatzierend, LRU-Rang
    } Priorisierung
  - Funktionalität vollständig mit dem Konzept der Co-Routine realisierbar

Typische Belegungssituation des Mini-I-Cache

Cache-Set	Weg 0	Weg 1
0	1	10
1	2	3
2	3	2
3		4
4	4	
5		
6	5	
7	6	
8	7	
9	8	11
10		
11		
12		
13		
14	9	
15		

Target-Routine	
1	HANDLER_BASE_HIGH
2	RESET_EXCEPTION_IS_DEBUG_HIGH
3	DYNAMIC_HANDLER_DOWNLOAD
4	RESET_EXCEPTION_IS_DEBUG_LOW
5	READ_MEM_32
6	READ_WORD
7	WRITE_WORD
8	HANDLER_FUNCTION_SWITCH
9	CLEAN_CACHES
10	HANDLER_BASE_LOW
11	RESTORE_CONTEXT
...	

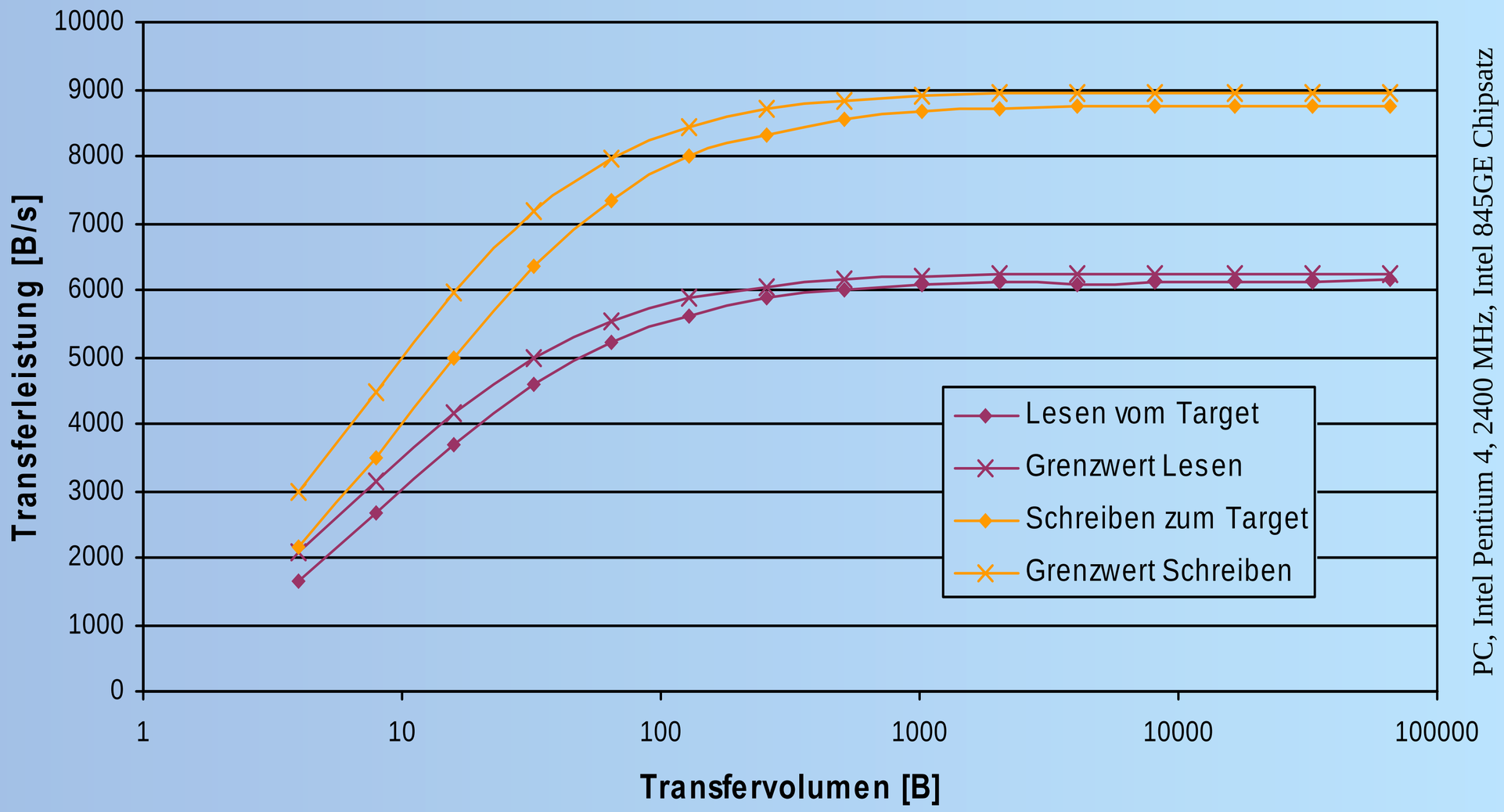
Softwareentwurf Targetinterface



Ausgewählte Aspekte der Implementation

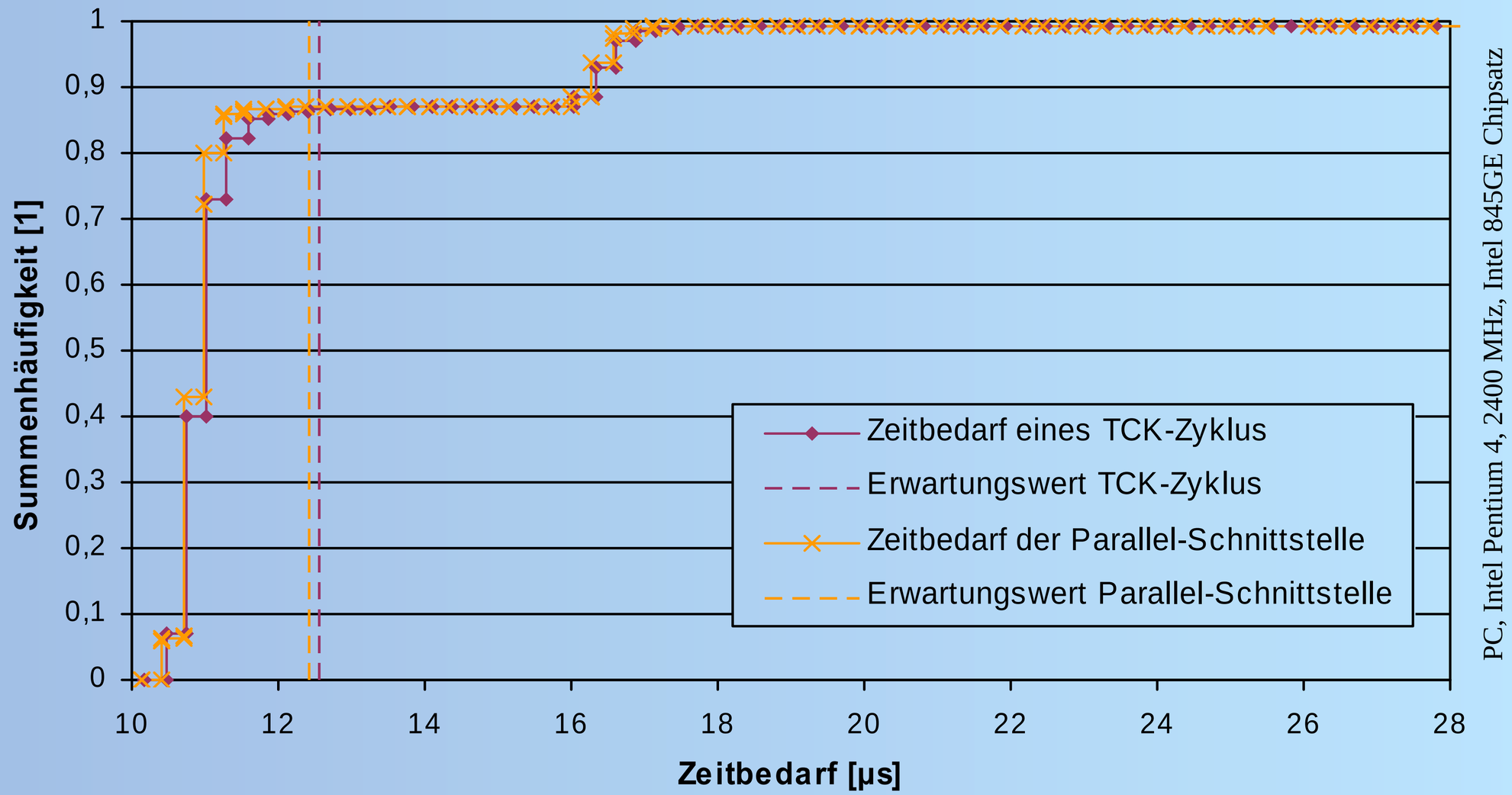
- Generator für TAP-Daten:
  - Modell des TAP-Kontrollers, Kopie des aktuellen TAP-Zustandes
  - Deterministische Suche einer Sequenzen von TMS-Daten
  - Speicherung bekannter Sequenzen der TMS-Daten
- Co-Routinen:
  - Polymorphe Realisierung
  - Einfacher, musterbasierter Linker mit Symboltabelle
  - Möglichkeit zur Selbstmodifikation einer Target-Routine
- Cache-Mapper:
  - Image-Technik
  - Platzierung der Target-Routinen im Mini-I-Cache:
    - Beachtung der Spezifika eines set-assoziativen Caches
    - Fragmentierung
    - Verdrängung von Target-Routinen
  - Prelink- und Link-Stufe
  - Differenzanalyse, Invalidierung und Download

- Transferleistung:



- Schreiben: zwei Hardwarezugriffe nötig ➔ max. 8,8 kB/s,
- Lesen: drei Hardwarezugriffe nötig ➔ max. 6,2 kB/s

- Zeitbedarf Targetinterface / Parallel-Schnittstelle:



- Mittelwert vollständiger TCK-Zyklus: 12,54 µs
- Mittelwert Schnittstellenzugriff: 12,40 µs

Zusammenfassung / Ausblick

- Zusammenfassung:
  - In-Circuit Debugging ist ein zentrales Mittel zur Verlässlichkeits- und Qualitätssicherung von Software auf eingebetteten Systemen
  - Generische Debug-Werkzeuge zeigen meist einen prozessor-spezifischen Anpassungsbedarf
  - Im Rahmen der vorgestellten Arbeit erfolgte die Anpassung der UDE an die Intel XScale Mikroarchitektur am Beispiel des Intel PXA 255
  - Das erstellte Targetinterface zeigt die nötigen Kompatibilitäts- und Flexibilitätseigenschaften, sowie eine ausreichende Leistungsfähigkeit
- Ausblick:
  - Erweiterung der Funktionalität:
    - Unterstützung der Trace-Hardware in Intel XScale Prozessoren
    - Realisierung einer Lösung für die Programmierung von FLASH-Speichern
    - Verbesserung der Unterstützung von Systemsoftware auf einem Target
  - Unterstützung weiterer Prozessoren aus der XScale Familie (IOP, IXP)
  - Abtrennung und Anpassung der ARM Core-Komponente (Version 5TE)
  - Realisierung eines „Universal Access Device“ für die XScale Architektur