TECHNISCHE UNIVERSITÄT DRESDEN

**Fakultät Informatik** Institut für technische Informatik, Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur

# Entwurf und Implementierung einer parametrierbaren Trace-Hardware am Beispiel der SHAP-Mikroarchitektur

Diplomverteidigung – 27. Januar 2010

Stefan Alex
s2174321@inf.tu-dresden.de

# Aufgabenstellung

- Literaturstudium zu Trace-Hardware in eingebetteten Systemen
- Analyse der Referenzszenarien im Hinblick auf notwendige Trace-Infrastruktur und deren Parametrierbarkeit.
- Evaluierung der Ansätze aus der Literatur bezüglich Eignung für die Referenzszenarien.
- Darauf aufbauend, Entwurf einer geeigneten allgemeinen Trace-Infrastruktur nebst Schnittstellen.
- Implementierung von Prototypen der Trace-Hardware-Module inklusive Module für die Vorverarbeitung.
- Test der Trace-Infrastruktur am Beispiel der SHAP-Mikroarchitektur mittels der Referenzszenarien
- Zusammenfassung und Dokumentation der Ergebnisse.

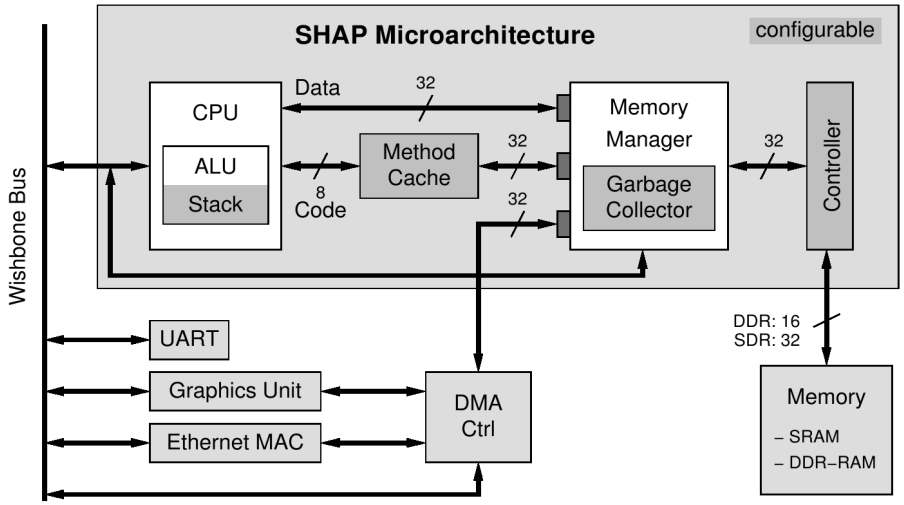
# Gliederung

1. Motivation
2. Anforderungen
3. Kompressionsverfahren
4. Architektur
5. Ergebnisse
6. Zusammenfassung und Ausblick

# 1. Motivation

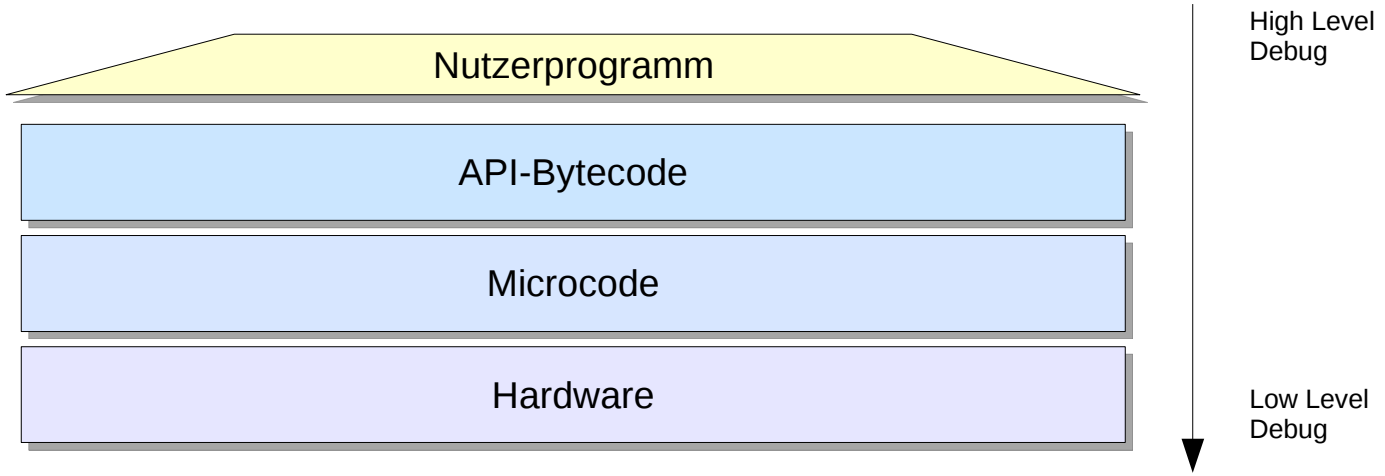
- Klassischer Ansatz – *In-Circuit Emulator*
  - Breakpoints/Single-Step
- Anhalten des Prozessors unter Umständen nicht sinnvoll
  - Beschädigung mechanischer Systeme
  - Maskierung des Fehlers
- Lösung – *Tracing*
  - Protokollierung interner Abläufe
  - Übertragung der Daten während oder nach der Aufzeichnung → Analyse/Fehlersuche auf Host-PC
- Profiling

# 2. Anforderungen
## SHAP



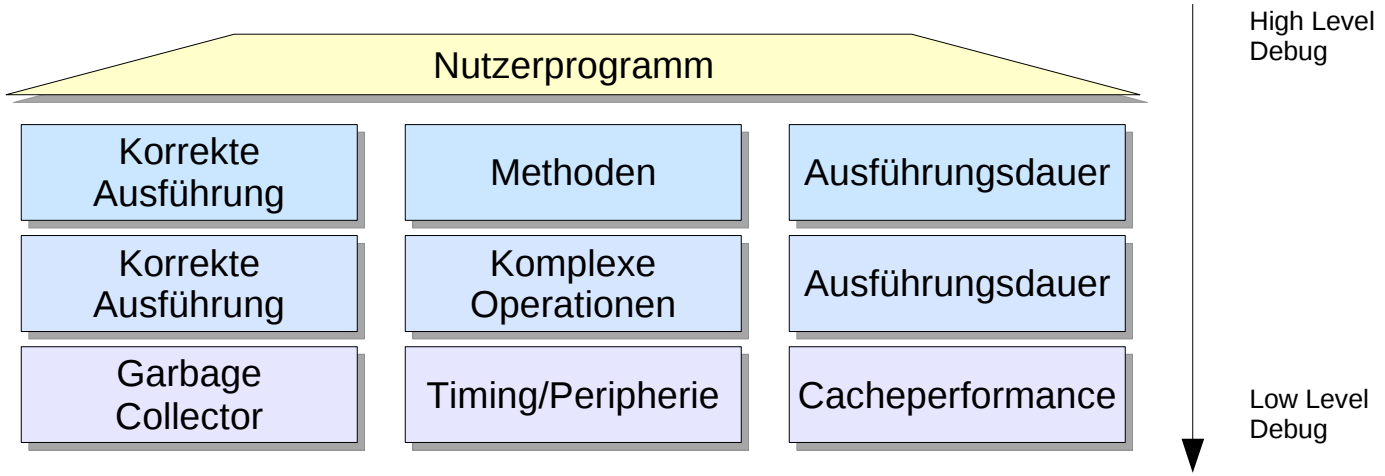

- Eingebetteter Bytecode-Prozessor
  - Bytecode löst Microcodebefehlsfolge aus
- Memory-Manager
  - Nebenläufiger Garbage-Collector
- Multi-Core

# 2. Anforderungen
## Referenzszenarien




- Verschiedene Anwendungsfälle → unterschiedliche Anforderungen

# 2. Anforderungen
## Referenzszenarien



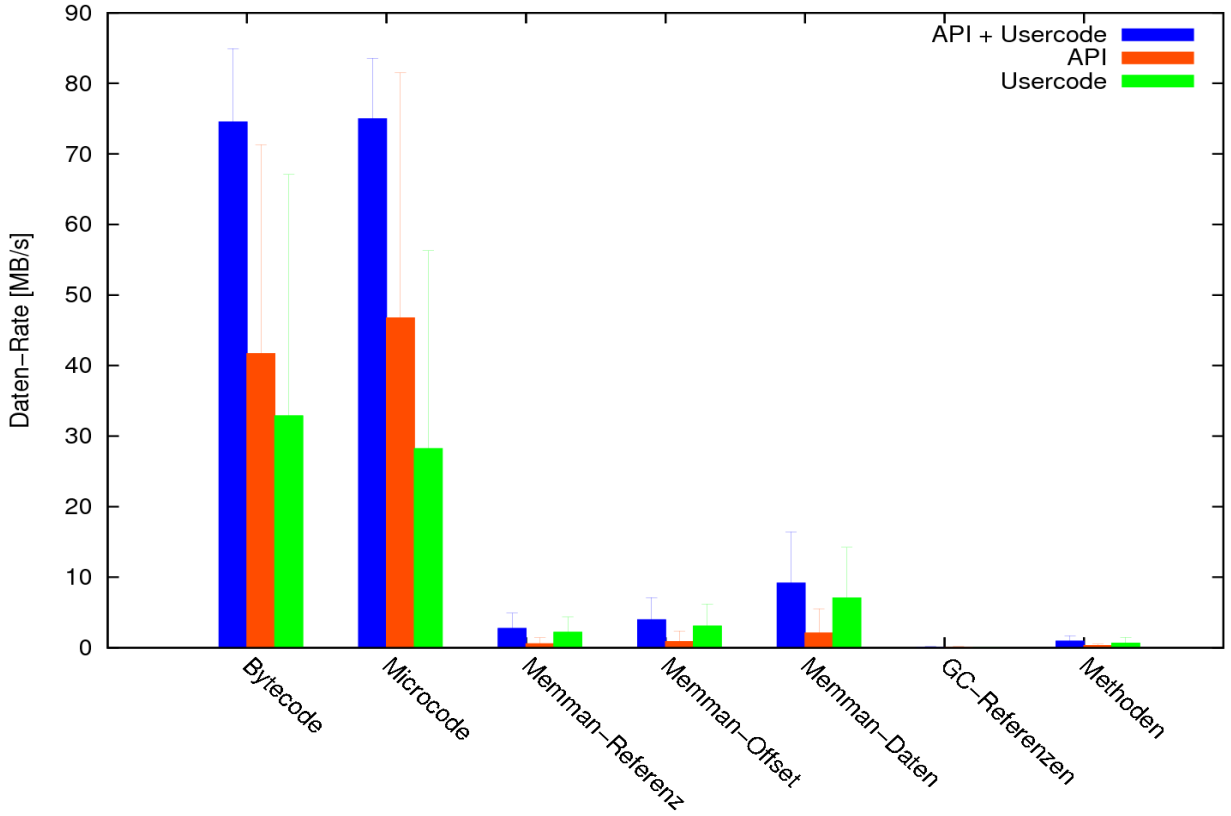

- Verschiedene Anwendungsfälle → unterschiedliche Anforderungen

# 2. Anforderungen
## Zielstellung

- Program-Trace
- Data-Trace
- Message-Trace
- Anzahl, Datenbreiten und Kombination der Trace-Module variabel
- Kompression
- Cross-Trigger
- Zyklusakkurater Trace
- Counter variabler Länge

# 3. Kompressionsverfahren

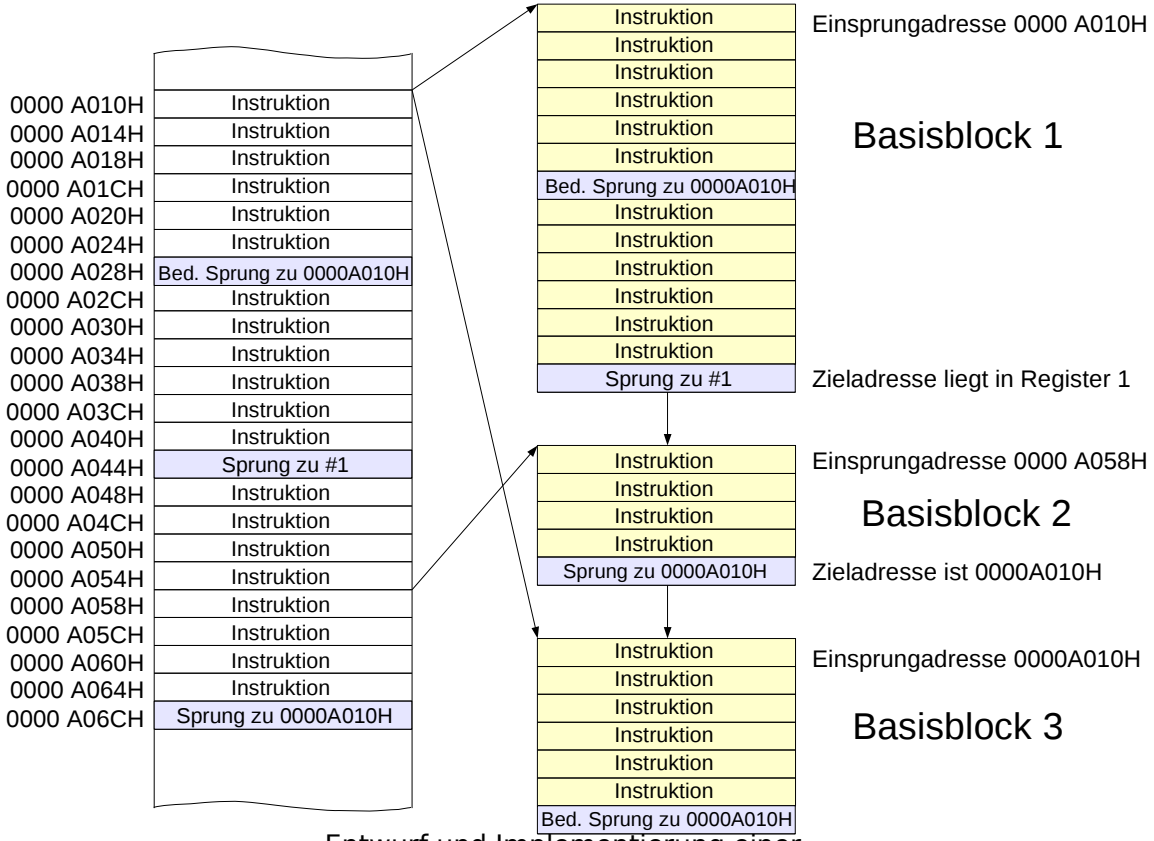
## Datenraten ohne Kompression



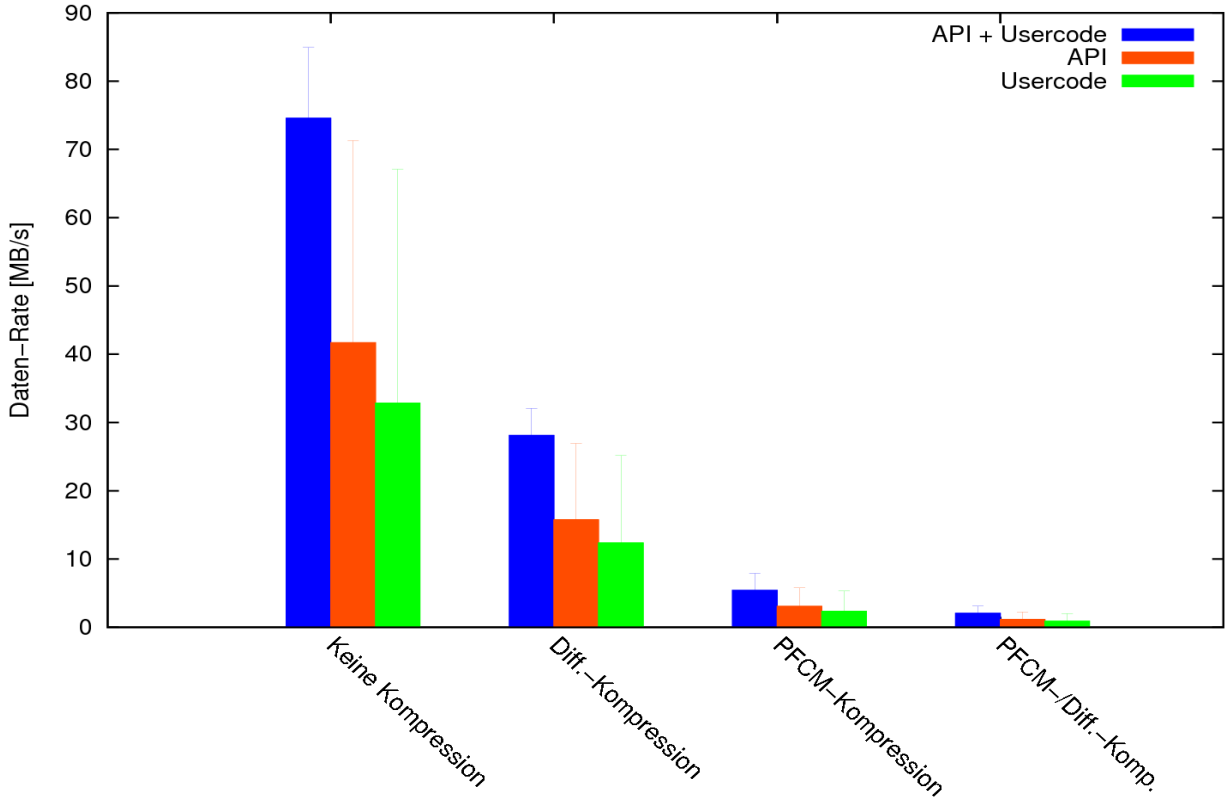
# 3. Kompressionsverfahren

- Lokalitätsbasierte Kompression
  - Differenz zum Vorgängerwert
  - Entfernen führender Nullen/Einsen
- Program-Flow-Change-Model
  - Kompression des Program-Trace
  - Unterteilung des Programmflusses in Basisblöcke
  - Übertragen der Basisblockidentifikation/Befehlszähler
  - Program-Image auf Host-PC

# 3. Kompressionsverfahren
## Program-Flow-Change-Model



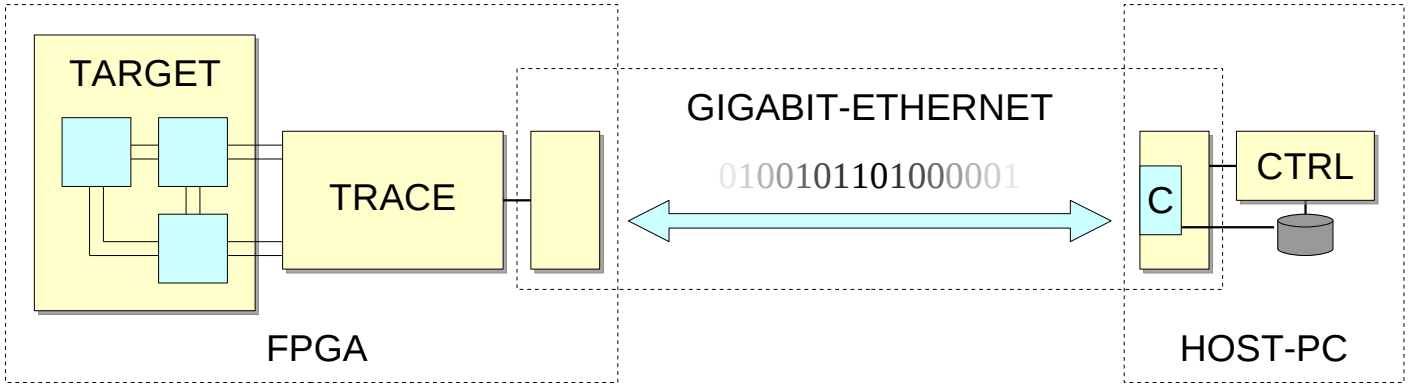
# 3. Kompressionsverfahren
## Bytecodekompression



# 3. Kompressionsverfahren
## Erweiterungen

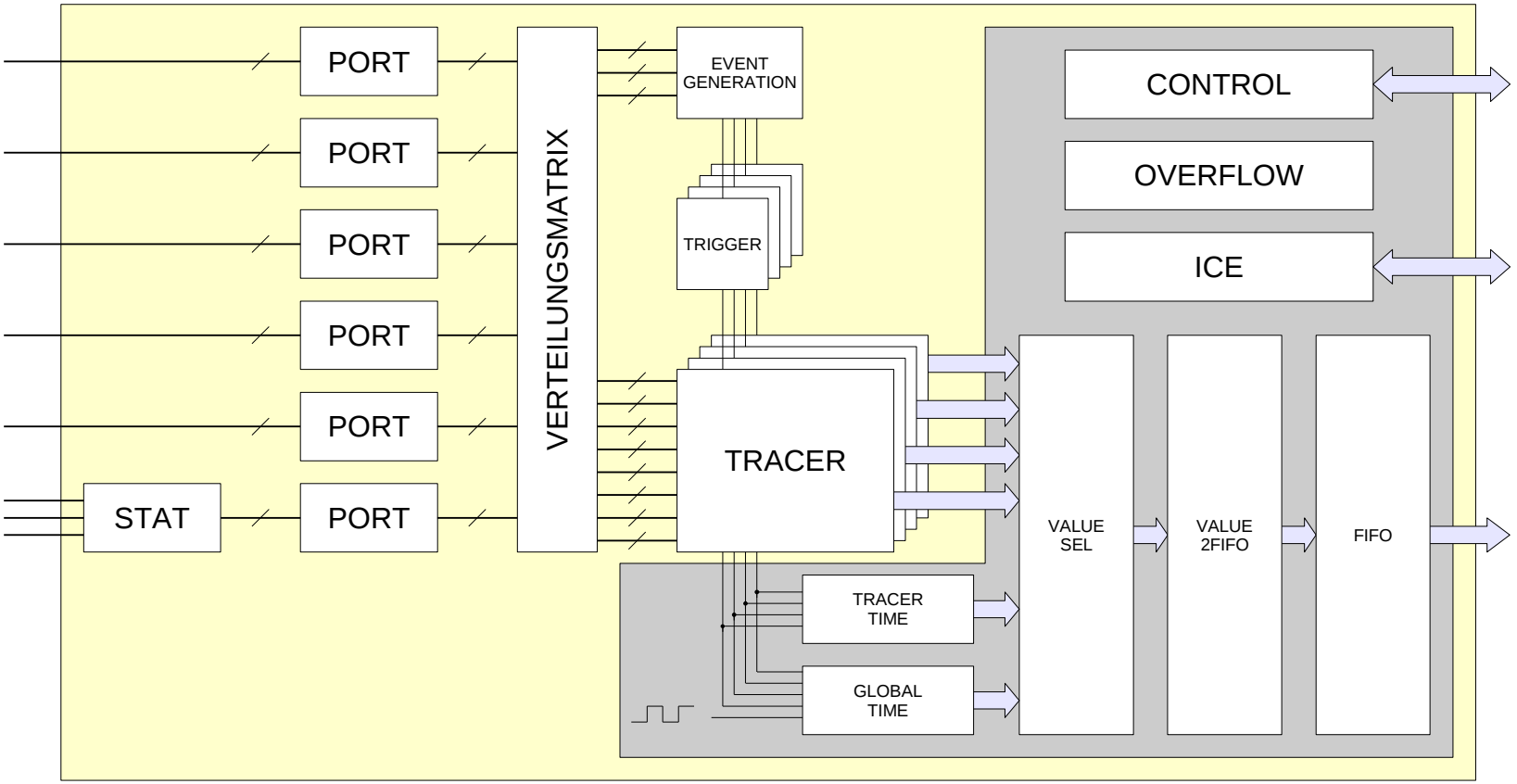
- Program-Flow-Change-Model
  - Inferieren der Zieladresse bei direkten Verzweigungen
  - History-Felder geben Ausführungsfolge direkter Verzweigungen an
    - Bitweise Kodierung
    - Längenkodierung
- Basisblockwiederholungen
- Generisches Kompressionsverfahren - Lempel-Ziv-77
  - Sliding Window
  - Verlust der Taktzuordnung

# 4. Architektur

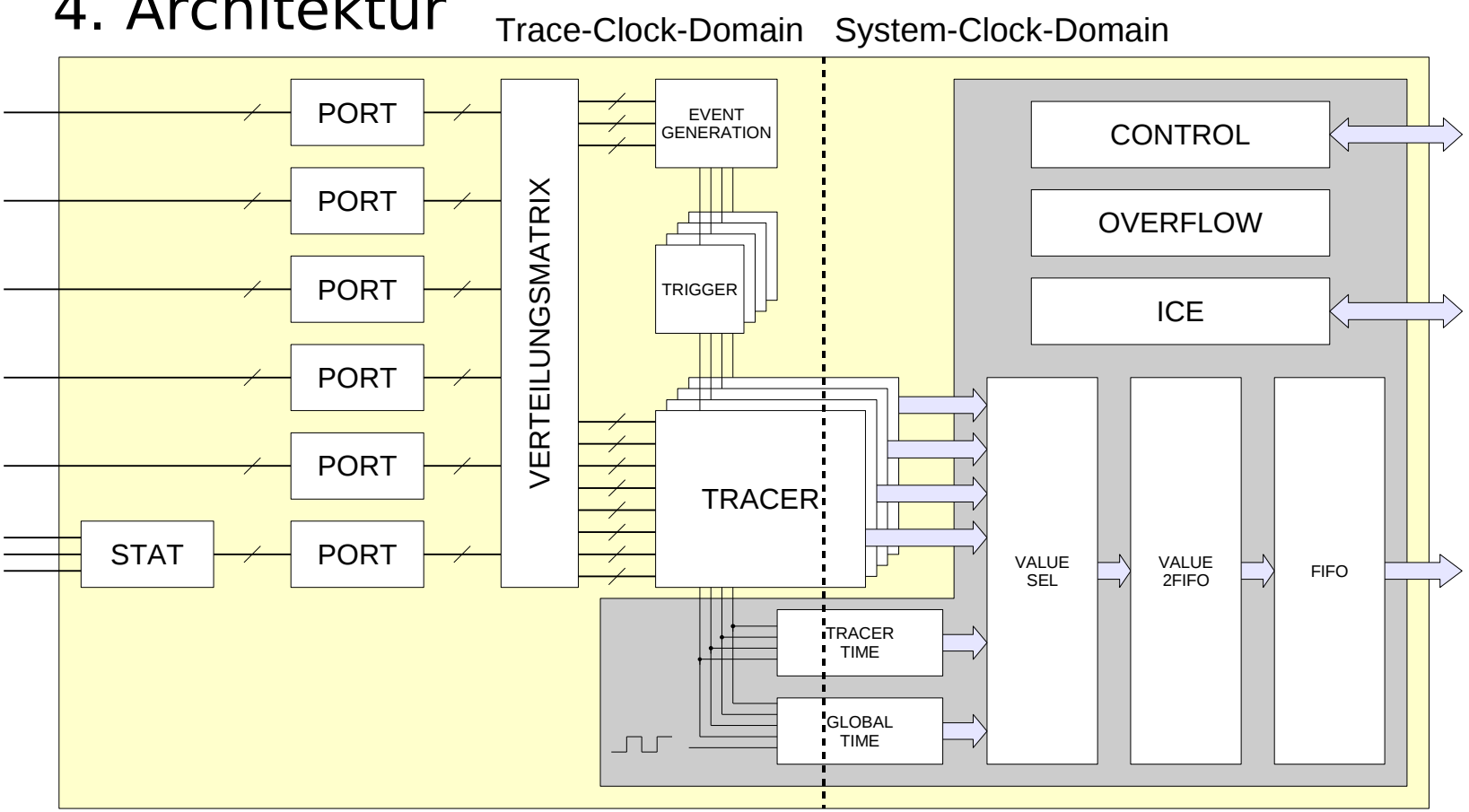



- Generische Schnittstelle
  - Off-/On-Chip-Trace möglich
- Realisierung als Off-Chip-Trace durch High-Speed-Schnittstelle des Prototyp-Boards

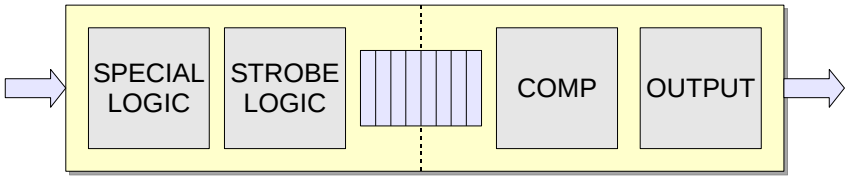
# 4. Architektur



# 4. Architektur



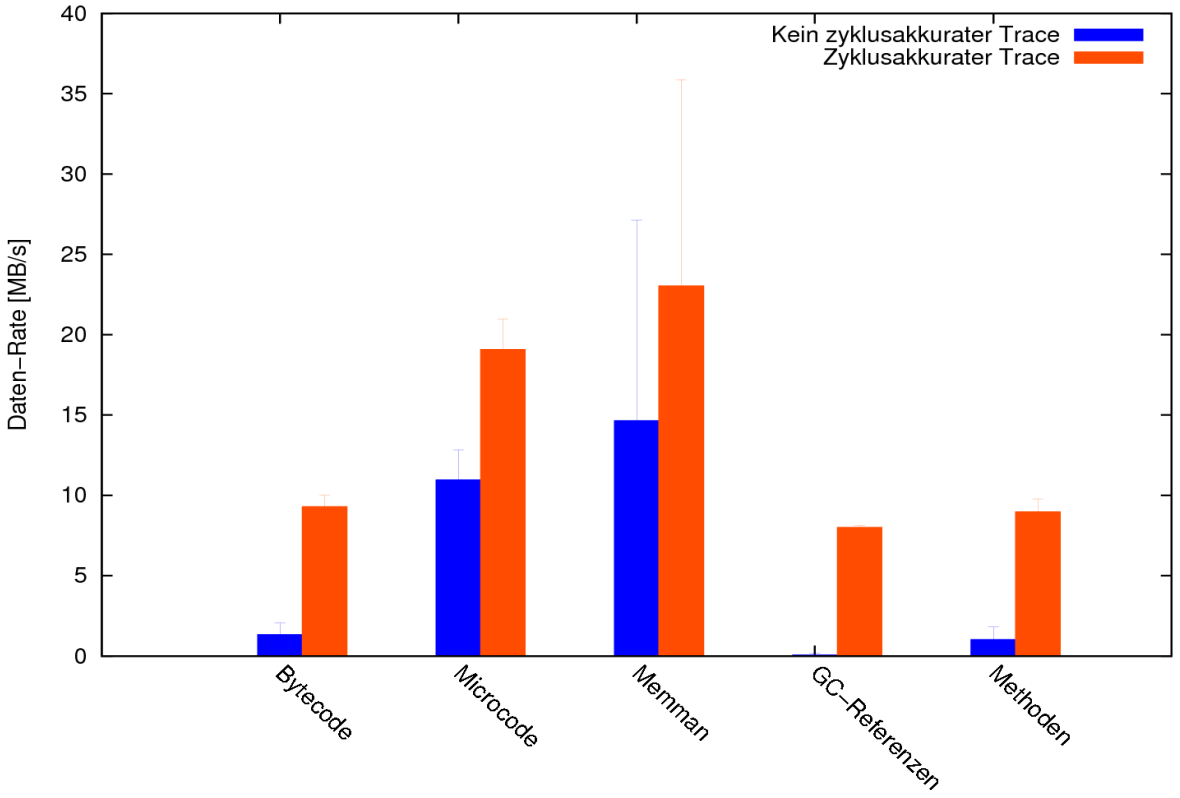
# 4. Architektur
## Tracer




- FIFO als Übergang zwischen Taktdomänen
- Generischer Nachrichtentrace
  - *Message-Tracer*
- Spezielle Implementierungen
  - *Memory-Tracer*
  - *Instruction-Tracer*
    - Program-Flow-Change-Model in verschiedenen Stufen

# 5. Ergebnisse

- Testanwendungen
  - FScript
  - HeapSort
  - IDEA-Crypt
  - Ray-Tracer
  - Sudoku



# 5. Ergebnisse
## Hardwareaufwand

- Vergleich zu SHAP
- Trace-System mit Gigabit-Ethernet
- Ein-Kern-System
- Bytecode-Trace: + *37,7* % LUTs, + *55,5* % Register
- Daten-Trace: + *50,1* % LUTs, + *74* % Register
- Methoden-Trace: + *37,4* % LUTs, + *52,2* % Register

# 6. Zusammenfassung und Ausblick

- Evaluierung existierender Ansätze
- Analyse der Kompressionsverfahren am SHAP
- Entwurf der Trace-Architektur
  - Hohe Wahlfreiheit bei der Instanziierung
  - Skalierbarkeit
  - Zyklusakkurater Trace
- Weiterer Ausbau der Architektur
  - Ausbau der Software zur Programmflussdarstellung
  - Taktgenauigkeit der Post- und Centertrigger

VIELEN DANK

FÜR DIE

AUFMERKSAMKEIT !

# Quellen

[1] Preußer, T.B.; Zabel, M.; Reichel, P., "The SHAP Microarchitecture and Java Virtual Machine" Technical Report Fakultät Informatik, TU Dresden, 2007

[2] Hu, X.; Chen, S.; „Applications of On-chip Trace on Debugging Embedded Processor“, Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007

[3] Krüger, M, "Grundlagen der Kraftfahrzeugelektronik", Hanser, 2. Auflage, 2008, ISBN 978-3-446-41428-0

[4] Mayer, A.; Siebert, H.; Lipsky, C., "Multi-Core Debug Solution IP - SoC Software Debugging and Performance Optimization, A White Paper", IPextreme Inc., Infineon Technologies AG, 2007

[5] Hopkins, A. B. T., McDonald-Maier, K. D., "Debug Support Strategy for Systems-on-Chips with Multiple Processor Cores", IEEE Trans. Comput., Volume 55, Nr. 2, 2006, IEEE Computer Society

[6] "The Nexus 5001 Forum - Standard for a Global Embedded Processor - Debug Interfacen", IEEE- Industry Standards and Technology Organization (IEEE-ISTO), Edition 2.0, 2003

[7] Kao, C.-F.; Chen, H.-M.; Huang, I.-J., "Hardware-Software Approaches to In-Circuit Emulation for Embedded Processors", IEEE Design & Test, Volume 25, Nr. 5, 2008, IEEE Computer Society Press

# 4a. Architektur
# Nachrichtenformat – Zeitinformationen

| Wert | Bedeutung |
|---|---|
| 00 | Tritt nach einer Nachricht auf, im selben Takt folgt eine weitere |
| 01 | Eine Nachricht folgt in einem Takt |
| 10 | Eine Nachricht folgt in zwei Takten |
| 11 | Keine Nachricht innerhalb des Intervalls |

- Erweiterung auf 4 oder 8 Bit möglich
  - Kodierung länger Zeiträume zwischen Events