SHAP — Secure Hardware Agent Platform

Martin Zabel, Thomas B. Preußer, Peter Reichel
{zabel,preusser}@ite.inf.tu-dresden.de,
Peter.Reichel@mailbox.tu-dresden.de

Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur
Prof. Dr.-Ing. habil. Rainer G. Spallek
Institut für Technische Informatik, TU Dresden

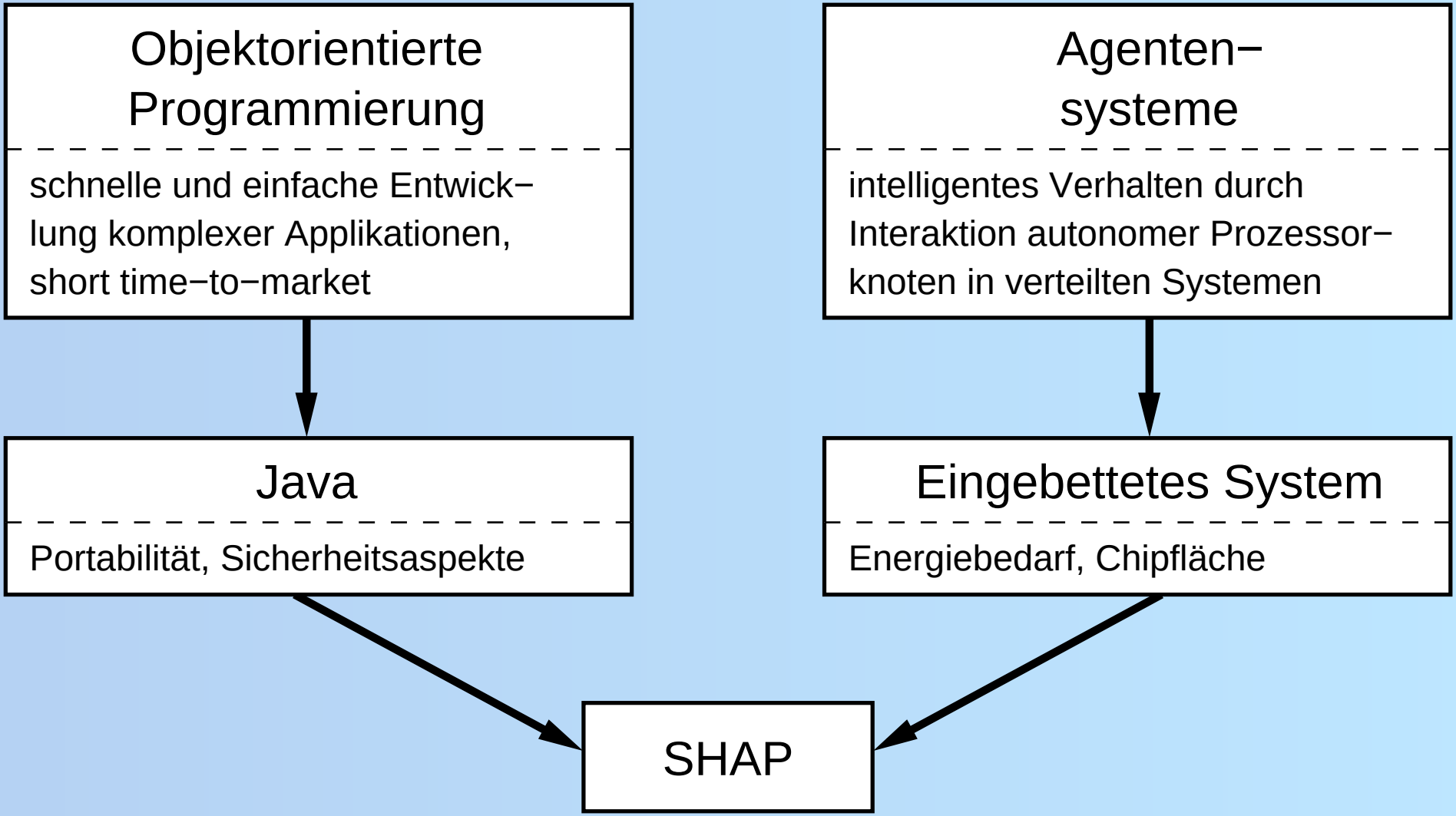
Gliederung

- Motivation
- SHAP-Konzept
- SHAP-Mikroarchitektur
- Zusammenfassung und Ausblick

Gliederung

- Motivation
- SHAP-Konzept
- SHAP-Mikroarchitektur
- Zusammenfassung und Ausblick

Ausgangspunkt



Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Ziele

Zielstellungen sind:

- eingebettetes System für Java-Applikationen
- Sicherheitsaspekte
- Echtzeitfähigkeit
- Multi-Threading
- Konzepte der Objektorientierung wie Exceptions, Garbage-Collection (GC), Interfaces, Serialisierung, dynamisches Nachladen von Klassen
- geringstmöglicher Hardwareaufwand

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Ähnliche Arbeiten

	FemtoJava	PicoJava-II	Berekovic	Komodo	JOP	JEM2 (aJ-100)	SHAP
Objects	-	+	+	+	+	+	+
Multiple Threads	-	+	-	(+)	(+)	+	+
Thread HW-Support	-	-	-	+	-	+	+
Scheduler	-	?	-	H	S	M	M
Synchronization	-	+	-	?	(+)	M	M
Real-time Threads	-	?	-	+	+	+	+
Garbage Collector	-	S	S	S	S	S	H
GC for Real-time Threads	-	?	-	+	-	-	+
Dynamic Class Loading	-	?	-	-	-	-	+

H	Hardware	M	Microcode	S	Software
+	supported	-	not supported	?	unknown

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

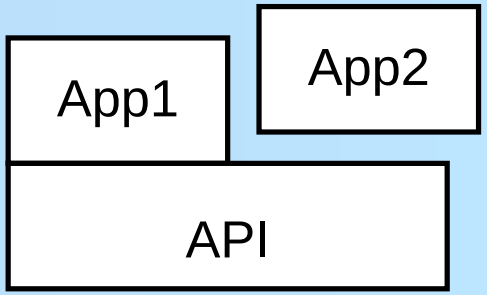
Gliederung

- Motivation
- SHAP-Konzept
- SHAP-Mikroarchitektur
- Zusammenfassung und Ausblick

Anwendungsszenarien

Denkbare Anwendungen von SHAP:

- Kombination mehrerer SHAP-Kerne zu einem Multi-Core-System mit gemeinsamen Speicher und automatischer Speicherverwaltung.
- Vernetzung mehrerer SHAPs zu einem Multi-Agenten-System inklusive Datentransfer auf Objektebene.
- Eingebettete Systemplattform, die neben der zugewiesenen Standardfunktionalität auch nachladbare nutzerbestimmte Java-Module (Nutzeragenten) unterstützt.



- Adaptiver filternder Monitorprozessor zur Systemüberwachung und Fehlerfahndung.

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Vorteile

SHAP bietet die Vorteile der Programmierung in Java basierend auf dem weitverbreiteten und akzeptierten API „Java Connected Limited Device Configuration“ (CLDC) mit der Unterstützung der High-Level-Programmierkonstrukte:

- Multithreading (mit zugesicherten Timeslots),
- Synchronisation durch Objekt-Monitore,
- automatische Speicherverwaltung,
- strukturierte Ausnahmebehandlung (Exception-Handling),
- Mehrfachvererbung durch Interfaces;

sowie das dynamische Nachladen von Java-Klassen (Applikationen).

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Echtzeitfähigkeit

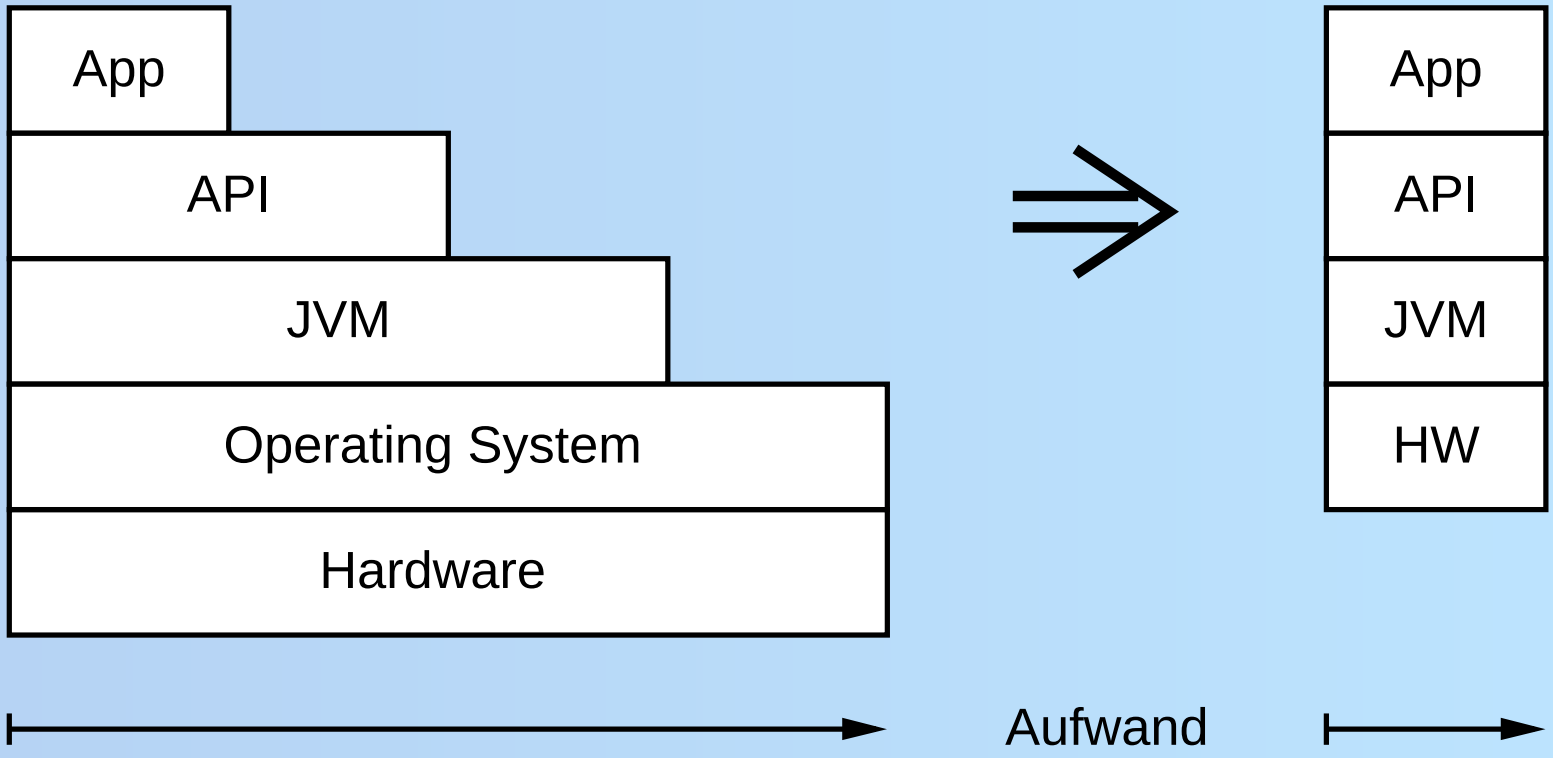
Echtzeitunterstützung durch:

- hardwareintegrierte Stack- und Threadverwaltung,
- hardwareintegrierte automatische, nebenläufige, d.h. prozessorunabhängige Speicherverwaltung inklusive Garbage-Collection,
- autonome Kontrollflüsse durch preemptives Round-Robin-Scheduling.

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Kein OS

Eignung für eingebettete Systeme, da kein Betriebssystem für die Ausführung der Java Virtual Machine erforderlich ist.



Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Gliederung

- Motivation
- SHAP-Konzept
- SHAP-Mikroarchitektur
- Zusammenfassung und Ausblick

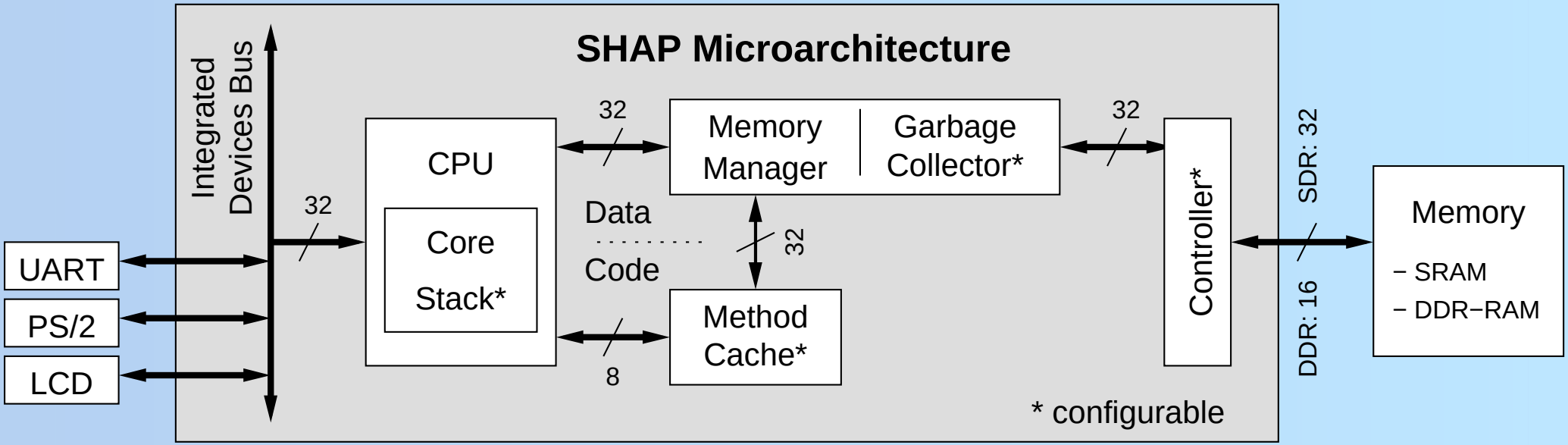
Überblick

Komponenten von SHAP:

- Hardware:
  - Mikroarchitektur, die direkt Java-Bytecode ausführt
  - externer Speicher für Java-Heap
  - verschiedene Busgeräte
- Software:
  - Implementierung der CLDC API
  - SHAP-Linker: Vorverarbeitung und Linken von `.class`-Dateien
  - Assembler für den Mikrokode der CPU

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Mikroarchitektur



HW-Komponenten der SHAP-Mikroarchitektur:

- CPU: Stackmaschine
- Memory Manager mit GC
- Method Cache[a]
- Integrated Memory Controller
- Integrated Devices Bus

[a] nur bei von-Neumann erforderlich

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Befehlssatz

SHAP unterstützt direkt Java-Bytecode mit folgenden Abweichungen:

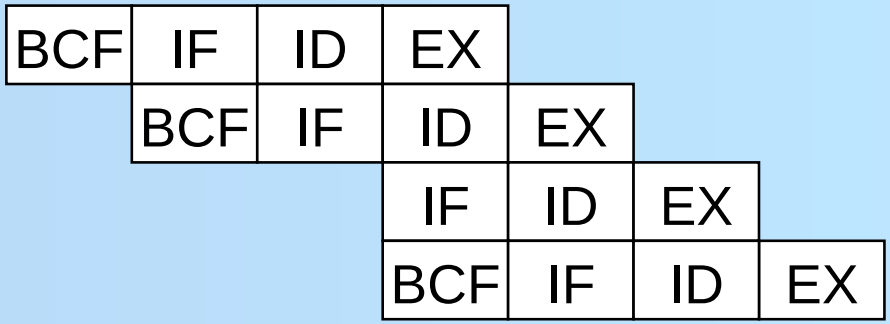
- modifizierte Bytecodes: Sprungbefehle mit absoluten Adressen,
- fehlende Bytecodes: Floating-Point-Arithmetik
- zusätzliche Bytecodes für Systemfunktionen: Thread-Handling, Zugriff auf Busgeräte, usw.,
- zusätzliche Bytecodes für Interfaces.

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Core

Eigenschaften des Cores:

- Abarbeitung der Java-Bytecodes durch Mikroprogramme
- 4-stufige Pipeline:



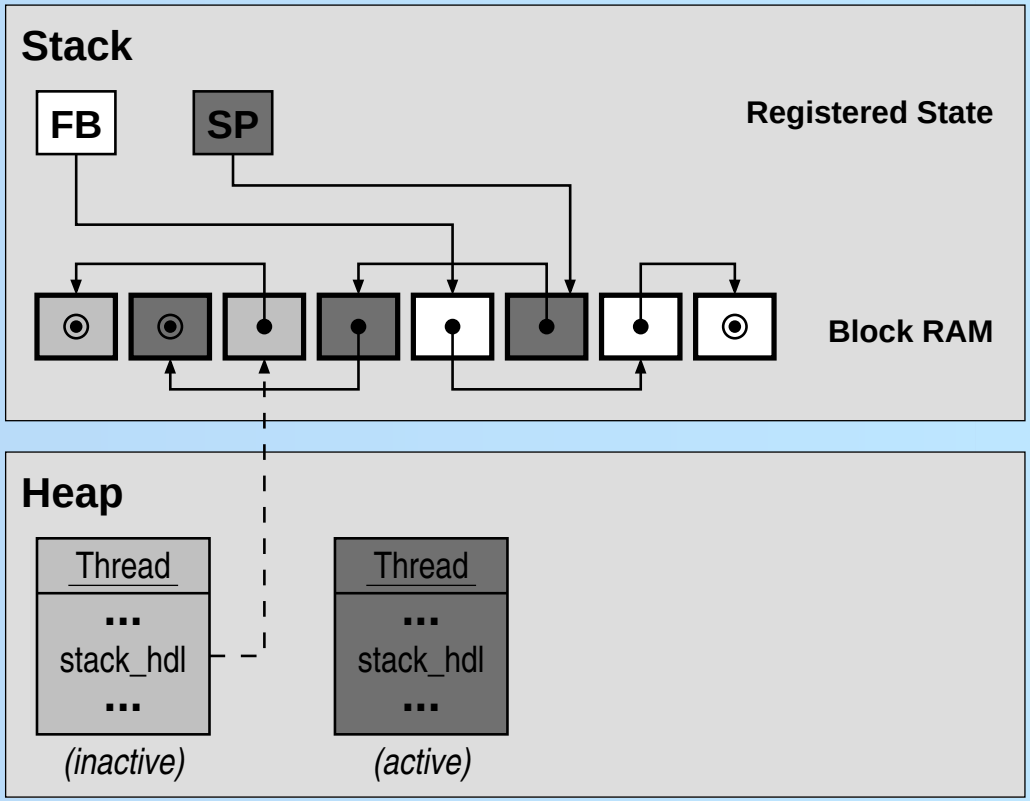
- interner 9-Bit-Befehlssatz,
  59 verschiedene Befehle
- Mikrokode-Datenspeicher, Werte zu 32 Bit
  - 16 Variablen
  - 48 Konstanten (an Stelle von Direktwerten)
  - 64 „Sprünge“

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Stack

Eigenschaften des Stacks:

- On-Chip-Speicher
- Unterteilung des Speichers in Blöcke gleicher Größe
- Verkettung dieser Blöcke zu einem Stack bzw. zur Freiliste
- eigenständiger Kontextwechsel
- Management der Methoden-Frames
- TOS und NOS in Register

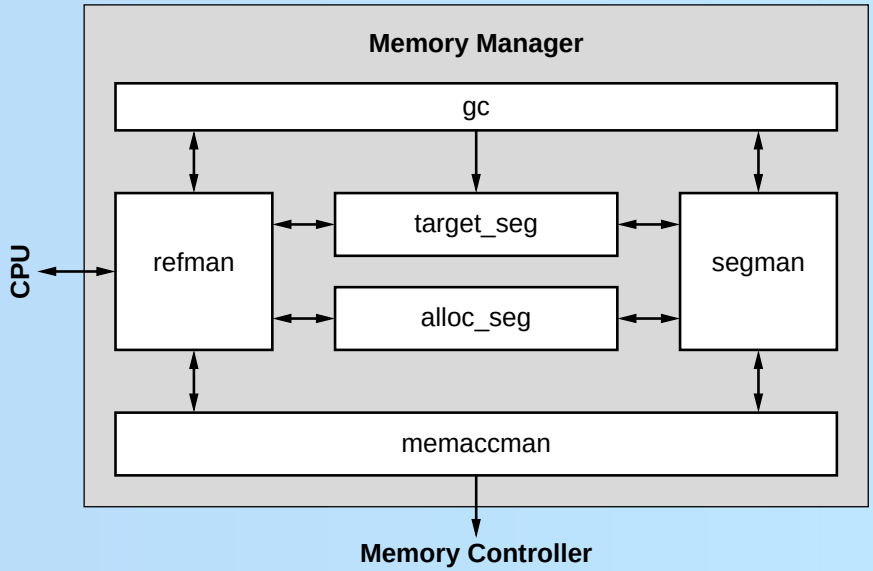


Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Memory Manager

Eigenschaften des Memory Managers:

- *Nebenläufig* zur CPU
- eigenständige Verwaltung von Objekten (Java-Heap), CPU arbeitet nur mit Referenzen und Offsets
- nebenläufiger Mark-and-Sweep-GC, Segmentierung des Hauptspeichers
- alle Operationen in konstanter Zeit



Zugriff auf den Hauptspeicher über Memory Controller, der eine einheitliche Schnittstelle für verschiedene Speichertypen zur Verfügung stellt.

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Scheduler

Eigenschaften des Schedulers:

- zur Zeit in Mikrokode implementiert
- Verwaltung von Threads mit Hilfe des Stack-Moduls
- preemptiver Round-Robin-Scheduler
- Implementierung der Synchronisation durch Objekt-Monitore
- blockierende Zugriffe auf Busgeräte
- Realisierung in Hardware möglich

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Prototyp

Prototyping auf einem SPARTAN3-Starter Kit Board (XC3S1000):

- 8 KByte Stack für bis zu 32 Threads
- 2 KByte Method Cache
- Memory Manager mit GC
- Taktfrequenz von 50 Mhz
- Memory Controller für externen 1 MByte SRAM
- Busgeräte: UART, LCD, PS/2, Memory Statistics Unit

	w/ GC		w/o GC	
Slices	2387	31%	1359	17%
Block RAMs	11	45%	8	33%
18×18 multipliers:	3	12%	3	12%
User I/O pins	95	54%	95	54%

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Gliederung

- Motivation
- SHAP-Konzept
- SHAP-Mikroarchitektur
- Zusammenfassung und Ausblick

Zusammenfassung

SHAP ist somit:

- Neuartige Implementierung einer eingebetteten Java-Mikroarchitektur für sichere, echtzeitfähige, und mehrfädige Applikationen.
- Mehrzweckplattform durch die Unterstützung von objektorientierten Konzepten wie: Ausnahmebehandlung, automatische GC und Interfaces.
- Implementierung der JVM in Mikrokode ohne darunter liegendem Betriebssystem.
- Echtzeitfähigkeit durch integriertes Stack- und Thread-Managment, nebenläufiger GC, autonome Kontrollflüsse durch preemptives Round-Robin-Scheduling.

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Ausblick

Weitere Arbeiten:

- Vervollständigung des dynamischen Nachladens von Klassen
- Untersuchung verschiedener GC-Strategien
- Hardware-Scheduler
- Multi-Core-SHAP

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

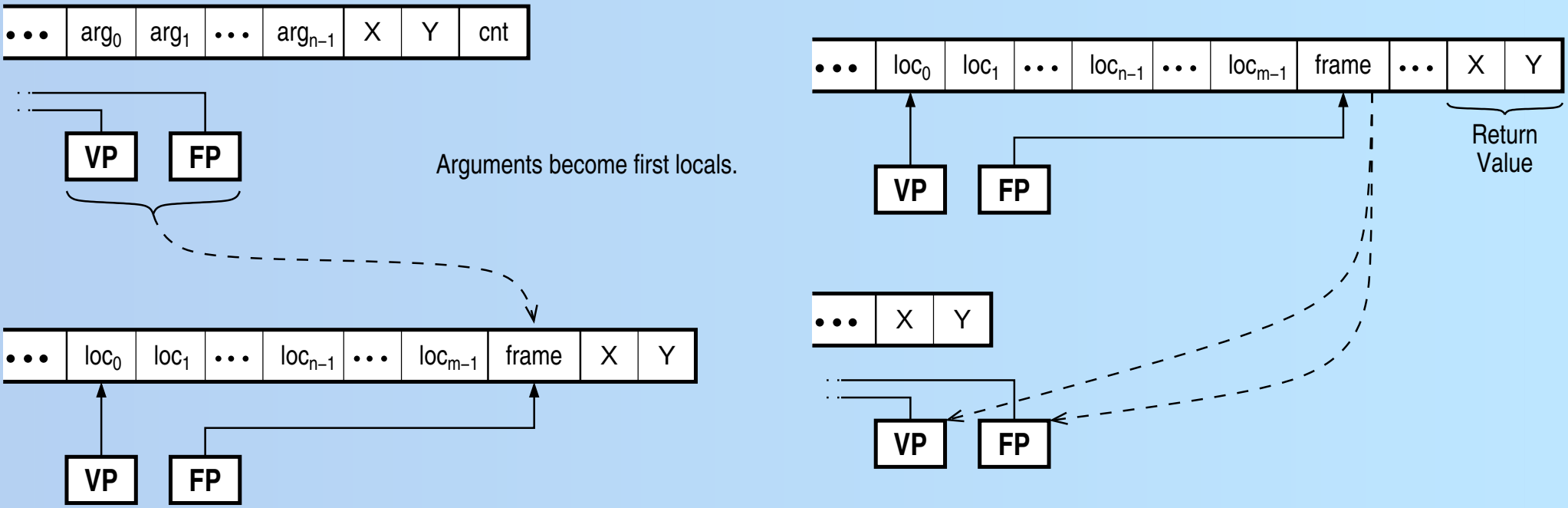
Vielen Dank für ihre Aufmerksamkeit.

Fragen?

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Methoden-Frames

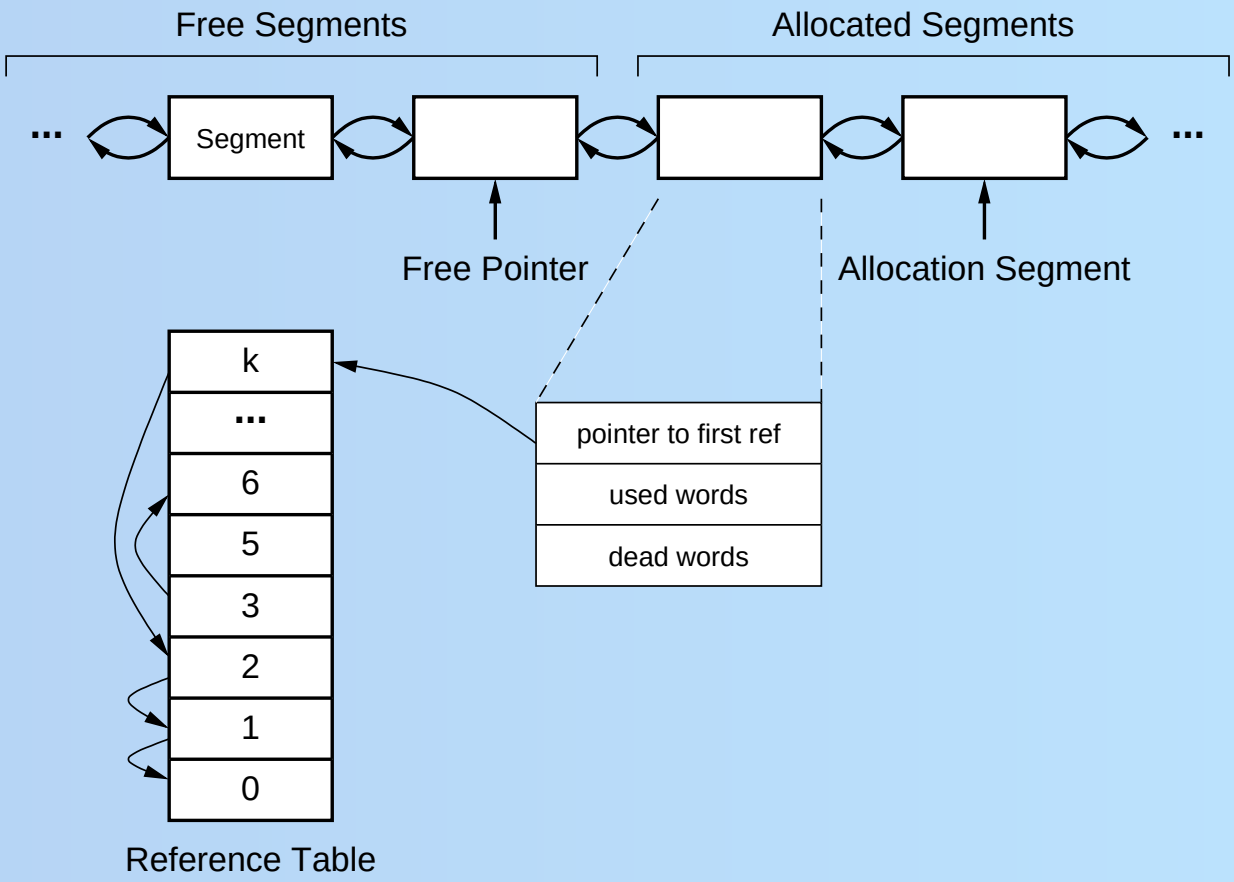
Auf- und Abbau von Methoden-Frames:



Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Heap-Segmente

Segmentierung des Heap-Speichers:



Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

JVM Implementierung

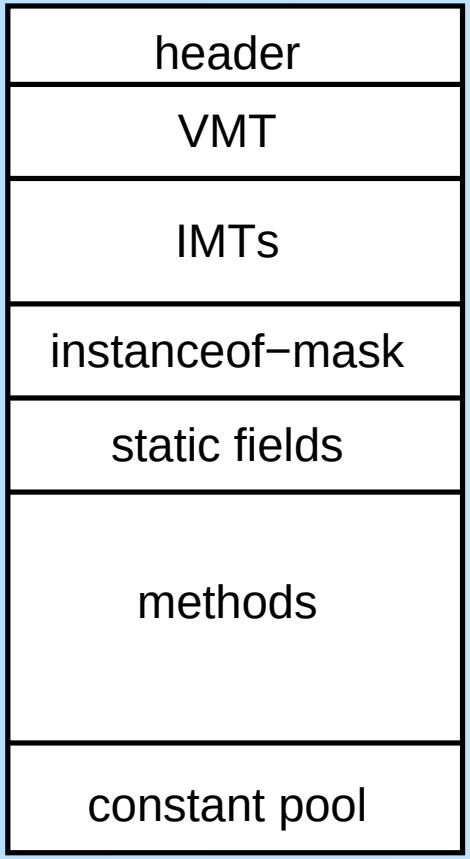
Die Java Virtual Machine ist komplett in Mikrokode implementiert und unterstützt folgende Konzepte:

- dynamische Allokation von Objekten,
- mehrere Threads,
- Synchronisation durch Objekt-Monitore,
- Ausnahmebehandlung,
- blockierende I/O-Operationen,
- dynamisches Nachladen von Klassen (in Arbeit).

Klassenobjekte

Konzept der Klassenobjekte:

- ein Klassenobjekt je Java-Klasse
- repräsentieren Instanzen von `java.lang.Class`
- enthalten alle Informationen zu einer Klasse
- ermöglichen dynamisches Nachladen von Klassen



Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de

Initialisierung und Bootstrapping

Booten des Systems:

1. Initialisierung der JVM und Laden des Bootstrap-Class-Loaders per Mikrokode.
2. Ausführung des in Java programmierten Bootstrap-Class-Loaders, der alle weiteren Klassen nachlädt.
3. Starten des neuen Threads `Startup` durch den Bootstrap-Class-Loader .
4. `Startup` initialisiert Klassen und ruft danach die `main()`-Methode der Applikation auf.

Anstelle der Applikation kann zunächst auch nur der System-Class-Loader geladen werden.

Ressourcenverbrauch der SHAP-JVM

Bedarf an Ressourcen der Mikroarchitektur:

- 1969 Mikrokode-Befehle
- 14 von 16 Mikrokode-Variablen
- 33 von 48 Mikrokode-Konstanten
- 58 von 64 Einträgen in der Sprungtabelle

Martin Zabel, Thomas B. Preußer, Peter Reichel
Institut für Technische Informatik, TU Dresden
{zabel,preusser}@ite.inf.tu-dresden.de, Peter.Reichel@mailbox.tu-dresden.de