TECHNISCHE UNIVERSITÄT DRESDEN

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

Anforderungsanalyse für Daten-Caches für die SHAP-Mikroarchitektur

Vortrag zum Beleg – 30. Juli 2008

Stefan Alex
s2174321@inf.tu-dresden.de

Gliederung

1. Aufgabenstellung
2. Caches - Grundlagen/Organisationsstrukturen
3. Lokalitätsprinzip
4. Analytische Modelle
5. Vorgehensweise
6. Quellen

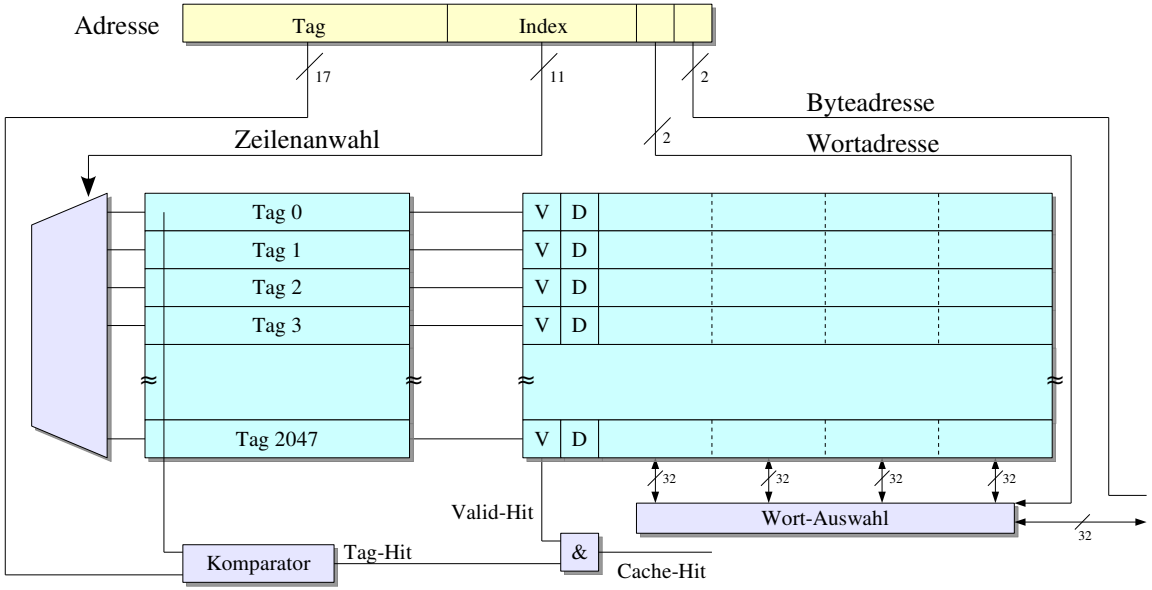
1. Aufgabenstellung

- Literaturstudium zu Strategien für Daten-Caches unter den Aspekten: Caching von Datenstrukturen (Objekte), Besonderheiten im Multi-Core-Bereich und praktischer Anwendung in eingebetteten Systemen.
- Entwurf von Protokollierungseinheiten für die Analyse der zeitlichen Lokalität von Objektaktivierungen einerseits und für die Analyse der zeitlichen und örtlichen Lokalität der Speicherzugriffe innerhalb der Objekte andererseits.
- Implementierung der Protokollierungseinheiten im Prozessorsimulator DITO als auch in Hardware (VHDL). Gegenseitiger Test anhand einer einfachen, dafür zugeschnittenen Testapplikation.
- Bestimmung der unter Punkt 2 genannten Kenndaten für mindestens 3 verschiedene, selbst gewählte Applikationen. Daraus ableitend, Formulierung von Anforderungen für den zukünftigen Entwurf eines TLB und eines Daten-Caches.
- Zusammenfassung und Dokumentation der Ergebnisse.

2. Caches - Grundlagen/Organisationsstrukturen

- Cache – französisch „cacher“ - verstecken
- häufig gebrauchte Speicherworte in prozessornahen Speicher
- erstmals vorgeschlagen 1965 von Wilkes
- erstmalig implementiert im IBM System 360/85 im Jahr 1968
- Ziel: hohe Bandbreite, großer Speicher, geringe Latenz, geringe Kosten

2. Caches - Grundlagen/Organisationsstrukturen
Assoziativität – Direct Mapped

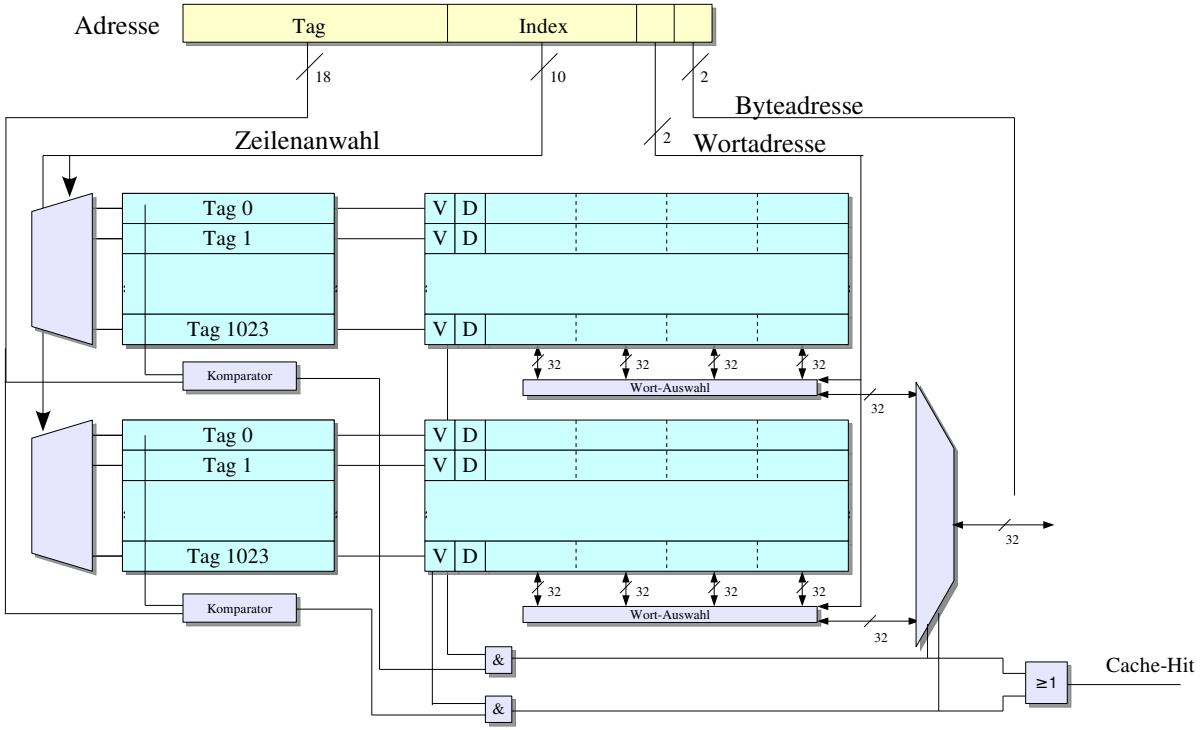


- Many-to-one-mapping der Speicheradressen in den Cache
- theoretisch große Blöcke in Cache speicherbar

Quelle: nach [7]

2. Caches - Grundlagen/Organisationsstrukturen

Assoziativität – Set associative

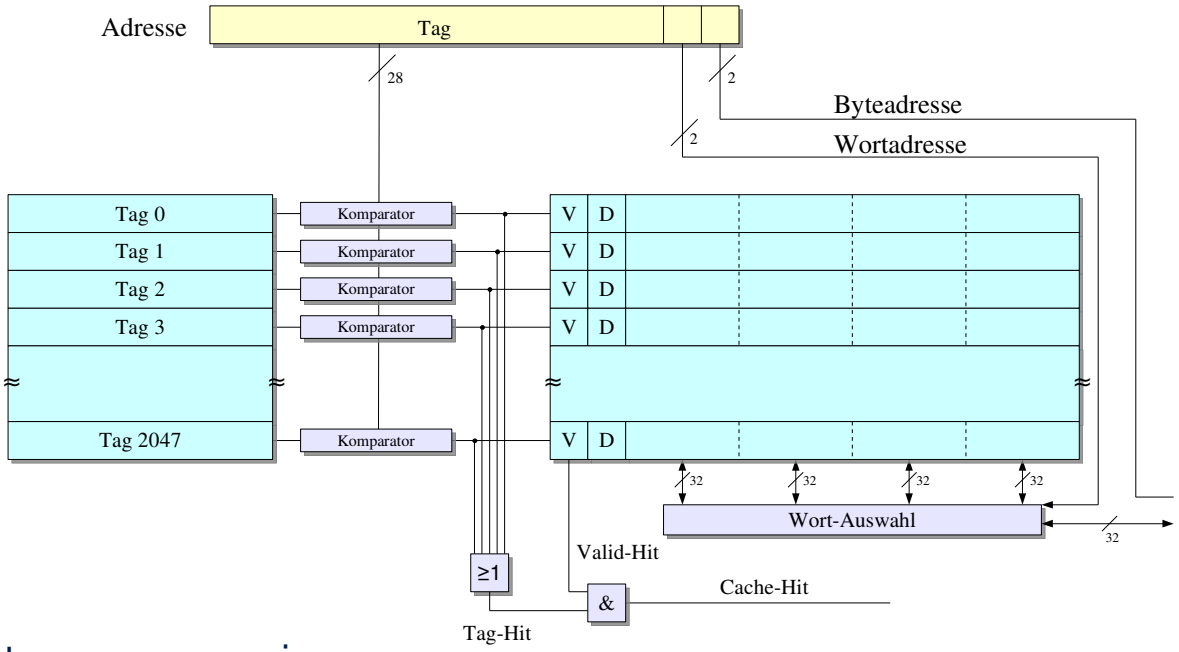


Quelle: nach [7]

- Many-to-few-mapping

2. Caches - Grundlagen/Organisationsstrukturen

Assoziativität – Fully associative



Quelle: nach [7]

- Many-to-many-mapping
- Vergleich mit allen Tags nötig / LRU schwierig zu implementieren

2. Caches - Grundlagen/Organisationsstrukturen
Verdrängungsstrategien

- FIFO (First In, First Out)
- LRU (Least Recently Used)
- LFU (Least Frequently Used)
- NMRU (Not Most Recently Used)
- Round Robin
- Random

2. Caches - Grundlagen/Organisationsstrukturen
Strategien

- Schreibstrategien
  - Write-through
    - Write-allocate
    - Non-write-allocate
  - Copy-back
  - Mischvarianten
- Zugriffsstrategien
  - Look Aside
  - Look Through

2. Caches - Grundlagen/Organisationsstrukturen
Optimierungsstrategien

- zusätzliche Speicher
  - Victim Buffer
  - Write Buffer
  - Linefill Buffer
- Pseudo-assoziative Caches
- Profiling (Way Prediction, ...)
- alternative Implementierungen: Hash-Caches, Skew-Caches, Smart Caches,...

3. Lokalitätsprinzip
Zeitliche Lokalität

- temporal locality
- Daten werden nach kurzer Zeit wieder referenziert
- Ursachen: z.B. Programmschleifen, Hot-Spot-Variablen
- Konsequenzen: Daten möglichst lange im Cache halten
  - Cache-Größe, Assoziativität, Verdrängungsstrategie
- Metrik: Stack-Distanz

3. Lokalitätsprinzip
Stack-Distanz

- Analyse des Adresstraces
- Zuordnung einer Stack-Distanz zu jedem Zugriff
- Stack-Distanz von A: Anzahl unterschiedlicher Zugriffe seit letzten Zugriff auf A
- verschiedene Granularität: Cache-Line, Speicheradresse
- Ziel: Charakterisierung von Programmen, Klassifikation von Cache-Misses, Grundlagen für analytische Modelle

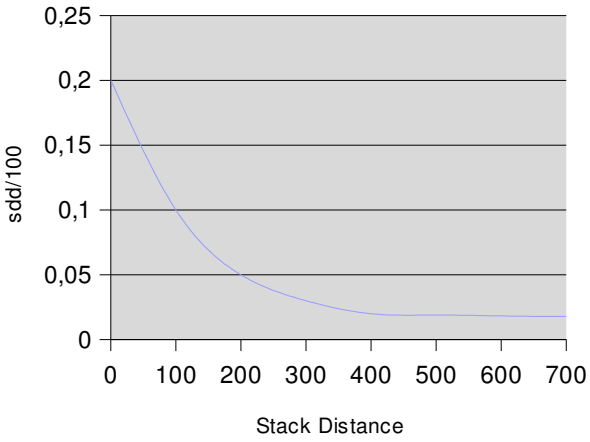
reference no.	1	2	3	4	5	6	7
address	0	8	16	96	8	16	104
corresponding cache line	0	0	1	6	0	1	6
stack distance (address gran.)	∞	∞	∞	∞	2	2	∞
stack distance (line gran.)	∞	0	∞	∞	2	2	2

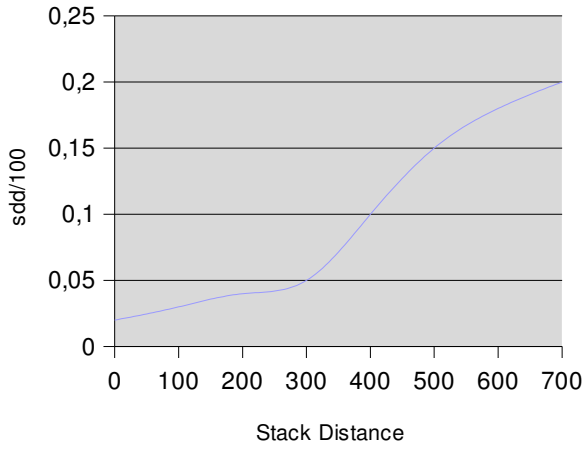
Quelle: [5]

3. Lokalitätsprinzip
Stack-Distanz

- Stack Distance Distribution

$$sdd(d) = \frac{(\sum_{t=0}^{T} eq(d, SD[t]))}{(T+1)}$$



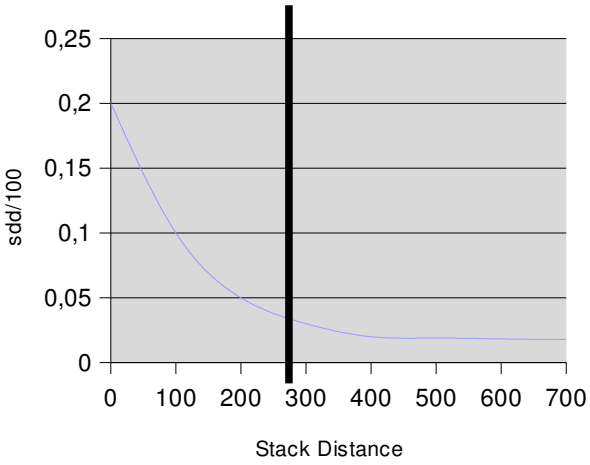


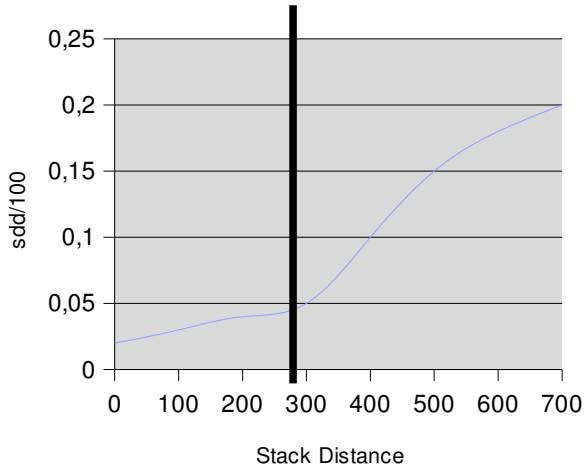
- Beispiele für hohe Lokalität (links) und niedrige Lokalität (rechts)
- Klassifikation von Cold-, Capacity- und Conflict-Misses (Three C's-Modell)

3. Lokalitätsprinzip
Stack-Distanz

- Stack Distance Distribution

$$sdd(d) = \frac{(\sum_{t=0}^{T} eq(d, SD[t]))}{(T+1)}$$





- Beispiele für hohe Lokalität (links) und niedrige Lokalität (rechts)
- Klassifikation von Cold-, Capacity- und Conflict-Misses (Three C's-Modell)

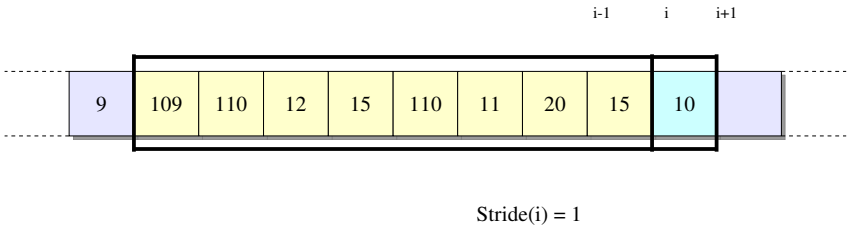
3. Lokalitätsprinzip
Örtliche Lokalität

- spatial locality
- Adressen „in der Nähe“ kürzlich verwendeter Daten werden aufgerufen
- Ursache: z.B. Datenstrukturen (Objekte, Arrays, Matrizen), Instruktionen
- Konsequenzen: neben eigentlichen Daten → Nachbardaten in Cache laden
  - Größe einer Cache-Line
- Metrik: verschiedene

3. Lokalitätsprinzip
Örtliche Lokalität - Metriken

- Markov-Modell
  - Berechnung von Wahrscheinlichkeiten für Durchläufe mit Schrittweite 1
  - durchschnittliche Länge der Durchläufe kann bestimmt werden

- „Stride“ eines Zugriffes (Block Distanz)
  - Abstand zu nächsten Nachbarn, der in bestimmtem Zeitfenster vorher aufgerufen wurde
  - Look Back Window



4. Analytische Modelle
Modell nach Agarwal, Horowitz & Hennessy

- Analyse des Adresstraces
  - Einteilen in „Time Granules“ (Zeitintervalle)
  - Bestimmen von Parametern pro Time Granule → Errechnen von Durchschnittswerten
- Klassifikation von Cache-Misses
  - Start-up misses
  - Nonstationary misses
  - Intrinsic Interference
  - Extrinsic Interference

4. Analytische Modelle
Modell nach Agarwal, Horowitz & Hennessy

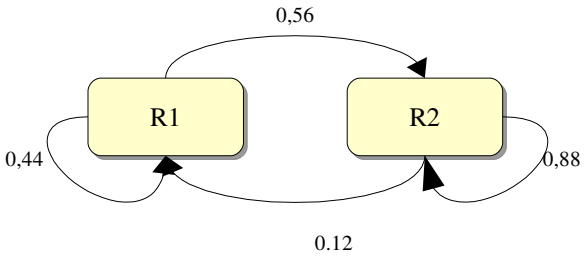
- Zielfunktion:

$$m(C,t)=\frac{u(B)}{\tau t}+\frac{U(B)-u(B)}{\tau T}+\frac{c}{\tau}\left[u(B)-\sum_{d=0}^{d=D} S\,d\,P(d)\right]$$

C	Cache-Organisation (S, D, B)
S	Anzahl an Sets
D	Grad der Assoziativität
B	Block-Größe
τ	Anzahl der Referenzen pro Time Granule
T	Anzahl an Time-Granules
t	Time-Granule
u(B)	durchschnittliche Anzahl an verschiedenen Speicherblockzugriffen pro Time Granule
U(B)	Anzahl unterschiedlicher Speicherblockzugriffe pro Trace
c	Kollisionsrate

4. Analytische Modelle
Modell nach Agarwal, Horowitz & Hennessy

- Örtliche Lokalität
- Darstellung durch 2-stufiges Markov-Modell
- Berechnung der Anzahl unterschiedlicher Zugriffe pro Time-granule auf Basis von Wahrscheinlichkeiten



$$u(B) = \frac{u(1)}{l_{av}} \sum_{l=1}^{l=\infty} R(l)\left(1 + \frac{l-1}{B}\right) = u(1)\frac{\left(1 + \left(f_{l1} / B\right) - f_{l2}\right)}{\left(1 + f_{l1} - f_{l2}\right)}$$

4. Analytische Modelle
Modell nach Agarwal, Horowitz & Hennessy

- Bestimmung der „Collision Rate“ durch repräsentative Messung im Adresstrace
- Annahmen:
  - Verteilung der Blöcke auf die Sets gleichwahrscheinlich
  - zufällige Ersetzungsstrategie
- Multiprogramming-Erweiterung:

$$m_i(C,t_i)_{extrinsic} = \frac{1}{t_i \tau} \lfloor \frac{(t_i - 1)}{t_s} \rfloor S \sum_{d=0}^{d=D} P_i(d) \sum_{e=D+1}^{e=u_{i'}(B)} \sum_{n=0}^{n=MIN(d,e)} \binom{e}{n} \left(\frac{d}{D}\right)^n \left(1-\frac{d}{D}\right)^{e-n} P_{i'}(e)$$

4. Analytische Modelle
Modell nach Brehob, Enbody

- Auswertung der Stack-Distanzen
- Wahrscheinlichkeit für jeden Zugriff, ob Datum im Cache

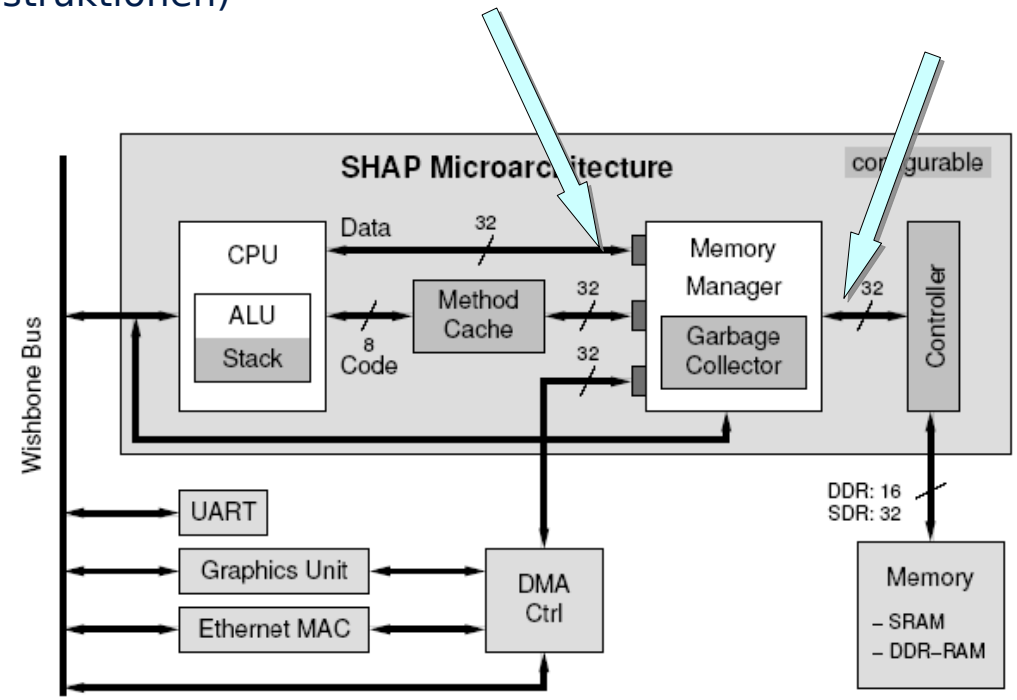
$$P_{HIT} = \sum_{a=0}^{a=A-1} \binom{D}{a} \left(\frac{A}{B}\right)^{a} \left(\frac{B-A}{B}\right)^{(D-a)}$$

- A Assoziativität
- D Stack Distance
- B Cache-Größe

- durchschnittlicher Fehler: 3 %
- wenige Parameter, Erweiterbar auf Örtliche Lokalität, Non-standard-Caches

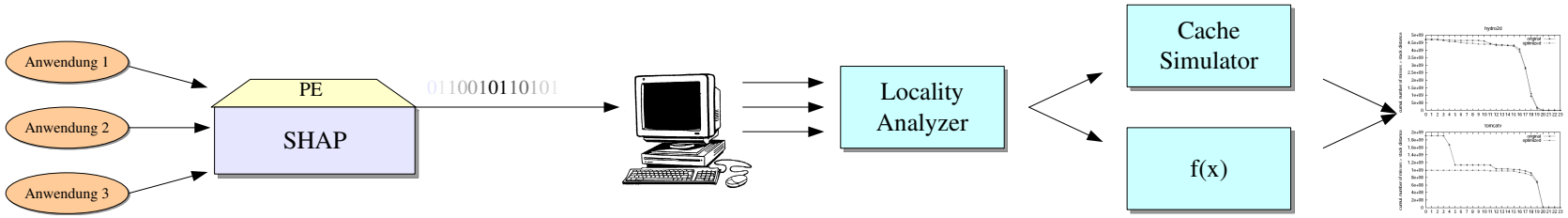
5. Vorgehensweise

- Implementierung von Protokollierungseinheiten
- Protokollieren von Zugriff, Adresse, Lese- oder Schreibzugriff, (Daten/Instruktionen)



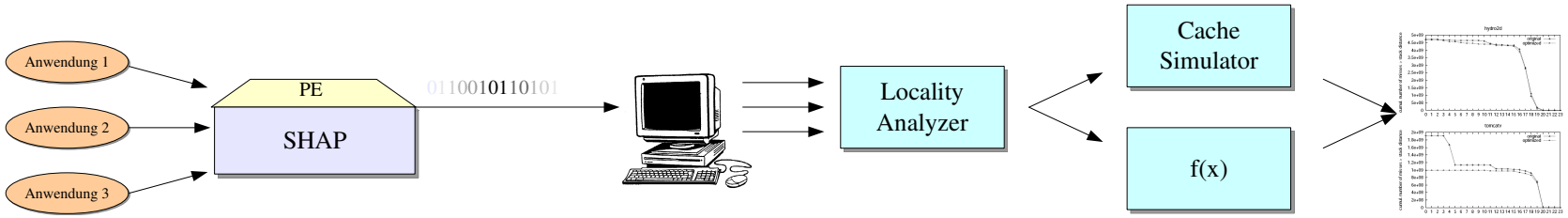
5. Vorgehensweise

- Aufzeichnen des Adress-Traces
- Übertragen an PC mittels UART/USB-Schnittstelle
- Charakterisierung und Auswahl geeigneter Testapplikationen
- Implementierung eines Cache-Simulators
- Analyse der Adress-Traces im Simulator für verschiedene Cache-Konfigurationen



5. Vorgehensweise

- Analyse der Lokalität → evtl. Anpassung der analytischen Modelle
- Implementierung der Zielfunktion, Vergleich mit Simulatorergebnissen
- Abschätzung des Hardwareaufwandes
- Ermittlung geeigneter Cache-Konfigurationen



6. Quellen

- [1] John L. Hennessy, David A. Patterson, „Computer Architecture – A Quantitative Appoach“, Morgan Kaufmann, 2003
- [2] Martin Zabel, Peter Reichel, Rainer G. Spallek, „Multi-Port-Speichermanager für die Java-Plattform SHAP“, 2008
- [2] Anant Agarwal, Mark Horowitz, John Hennessy, „An Analytical Cache Model“, Computer Systems Laboratory, Stanford University
- [3] Mark Brehob, Richard Enbody, „An Analytical Model of Locality and Caching“, Michigan State University, Department of Computer Science and Engineering
- [4] Jonathan Weinberg, Michael O. McCracken, Allan Snavely, Erich Strohmaier, „Quantifying Locality In The Memory Access Patterns of HPC Applications“, San Diego Supercomputer Center, University of California, San Diego
- [5] Kristof Beyls, Erik H. D'Hollander, „Reuse Distance as a Metric for Cache Behavior“
- [6] Cory J. Fox, James S. Gonzalez, Kelley C. Jones, „Studying the Quantitative Effects of Parameter Variation on Temporal and Spatial Locality“, Department of Computer Science, Florida State University
- [7] Prof. Dr.-Ing. Reinhold Orglmeister – Vorlesung Mikroprozessortechnik, www.emsp.tu-berlin.de/lehre/Mikroprozessortechnik/uebung/abbildungen11.pdf, TU Berlin, Fachgebiet Elektronik und medizinische Signalverarbeitung