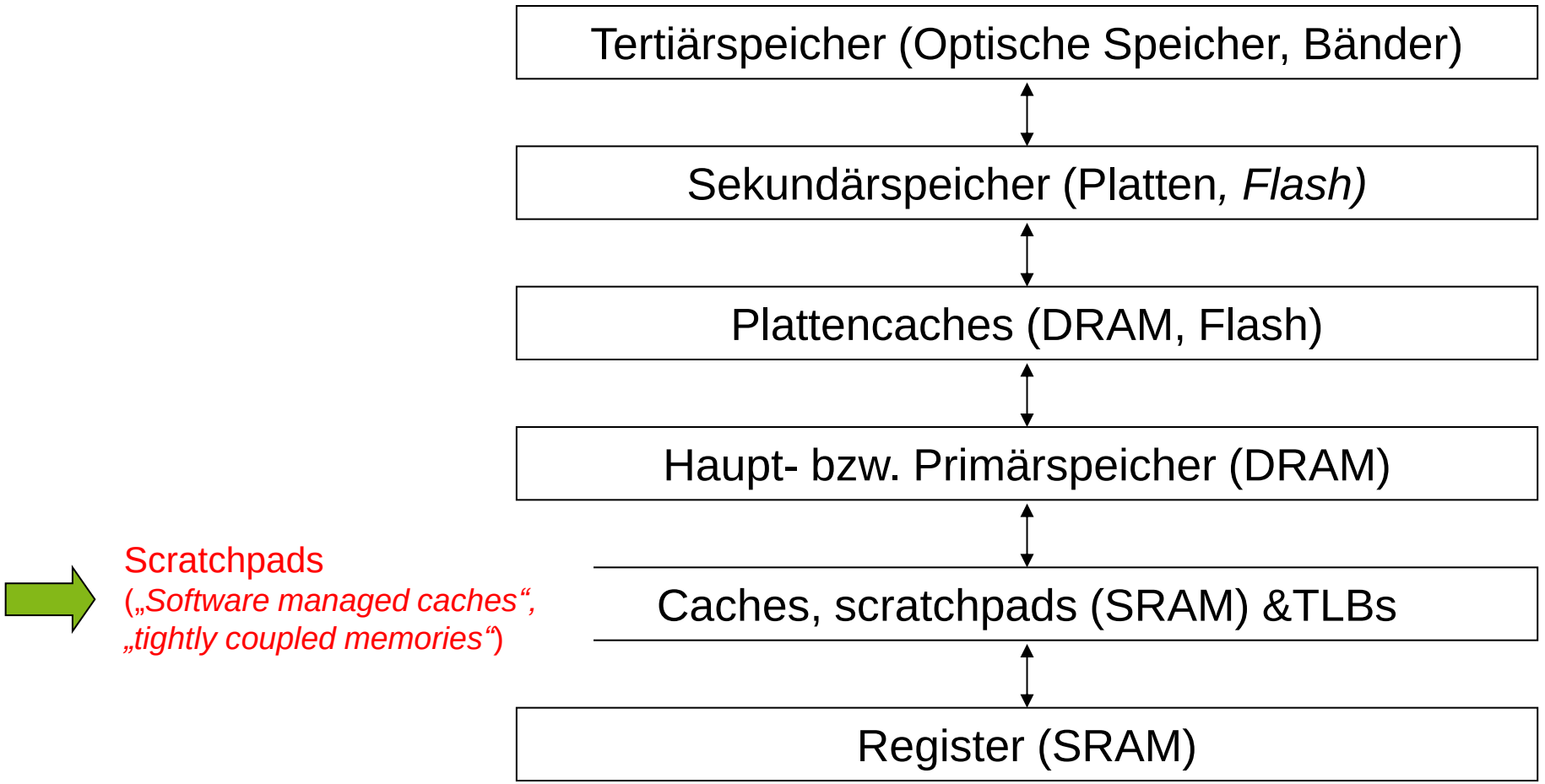
Speicherhierarchie: Scratch Pad- und Flash-Speicher

Peter Marwedel
Informatik 12

Mögliche Stufen der Speicherhierarchie
und derzeit eingesetzte Technologien



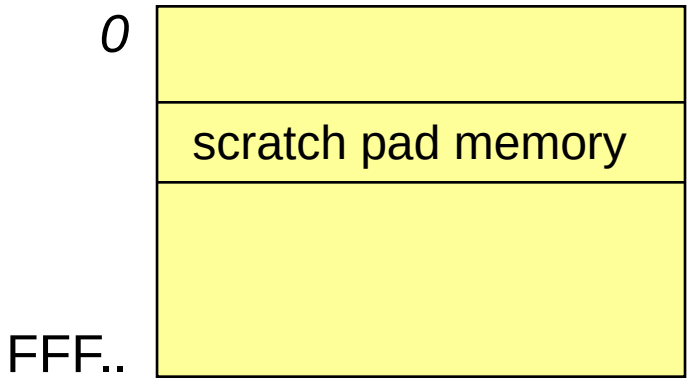
Scratch pad

Scratch pad?

© G. Marwedel, 2014

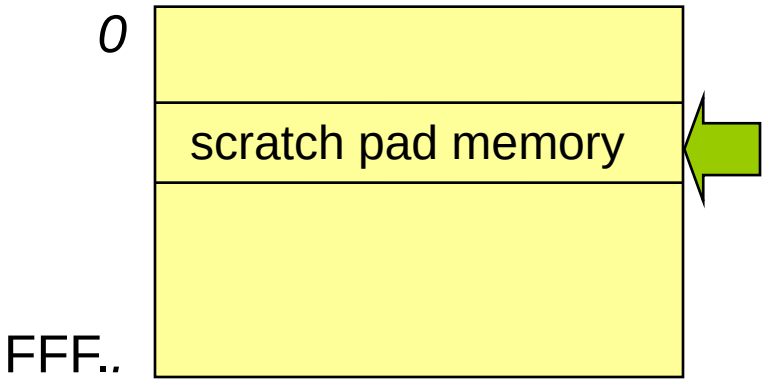
Scratch pad?

Scratch pad!
SPMs are small, physically separate memories mapped into the address space;



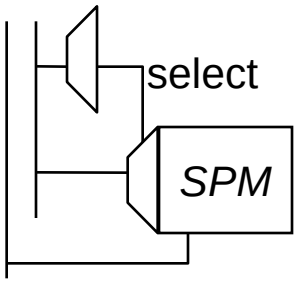
Scratch pad memories (SPM): Fast, energy-efficient, timing-predictable

Address space



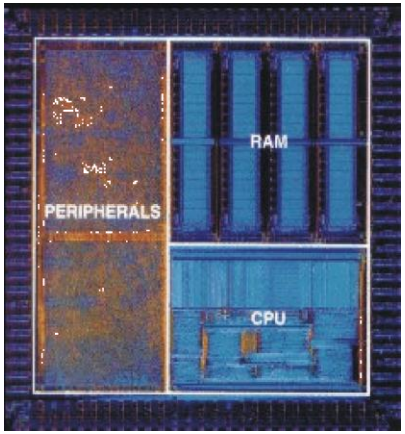
Small; no tag memory

Selection is by an appropriate address decoder (simple*!)*



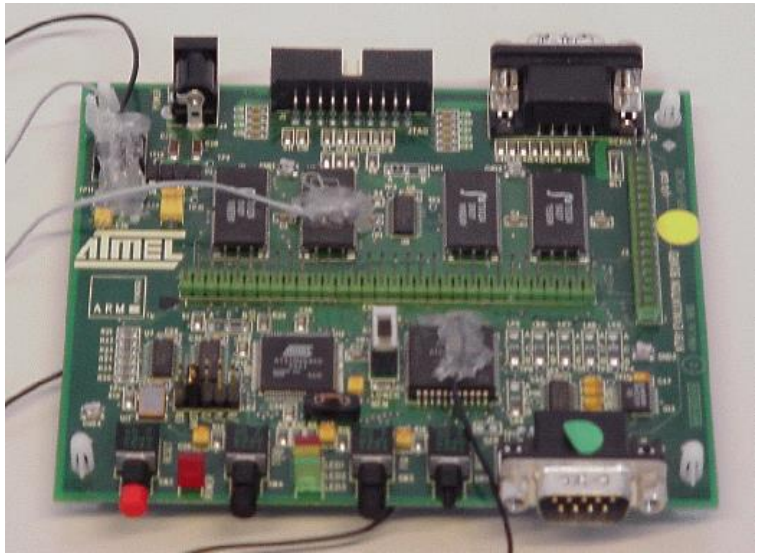
Example

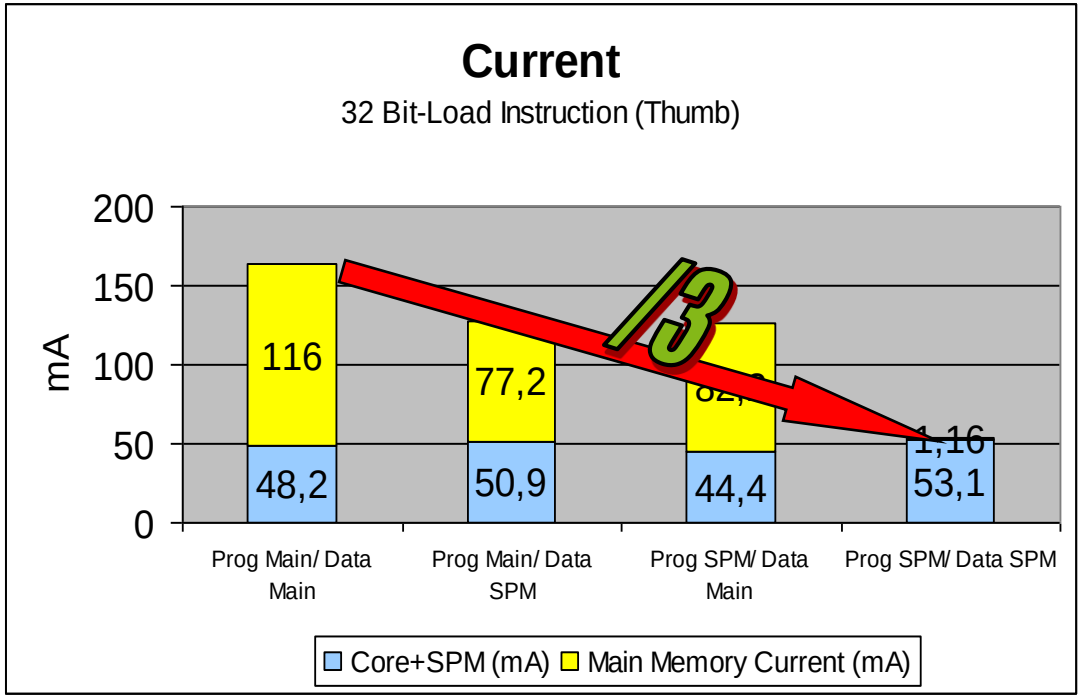
ARM7TDMI cores, well-known for low power consumption



Comparison of currents using measurements

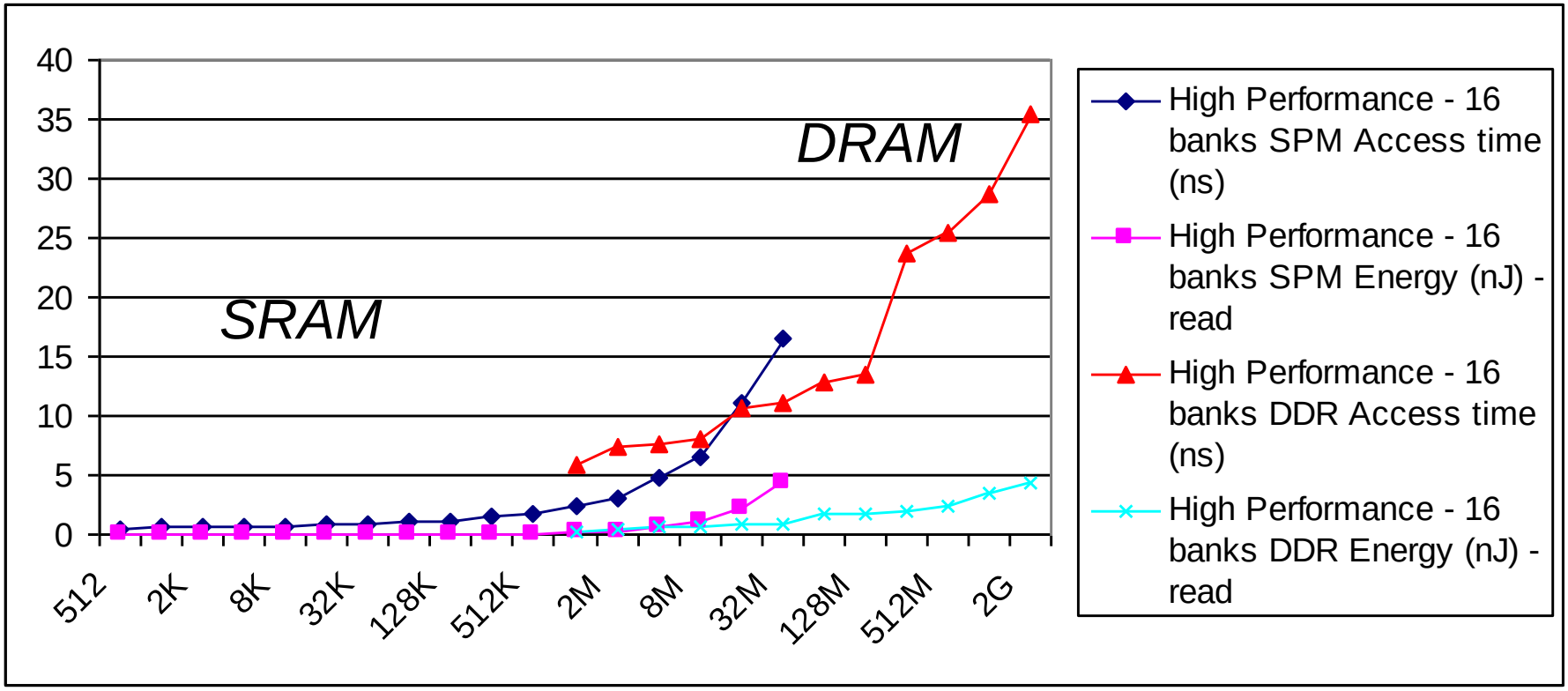
E.g.: ATMEL board with ARM7TDMI and ext. SRAM





Energy consumption of memories

Example CACTI: Scratchpad (SRAM) vs. DRAM (DDR2):

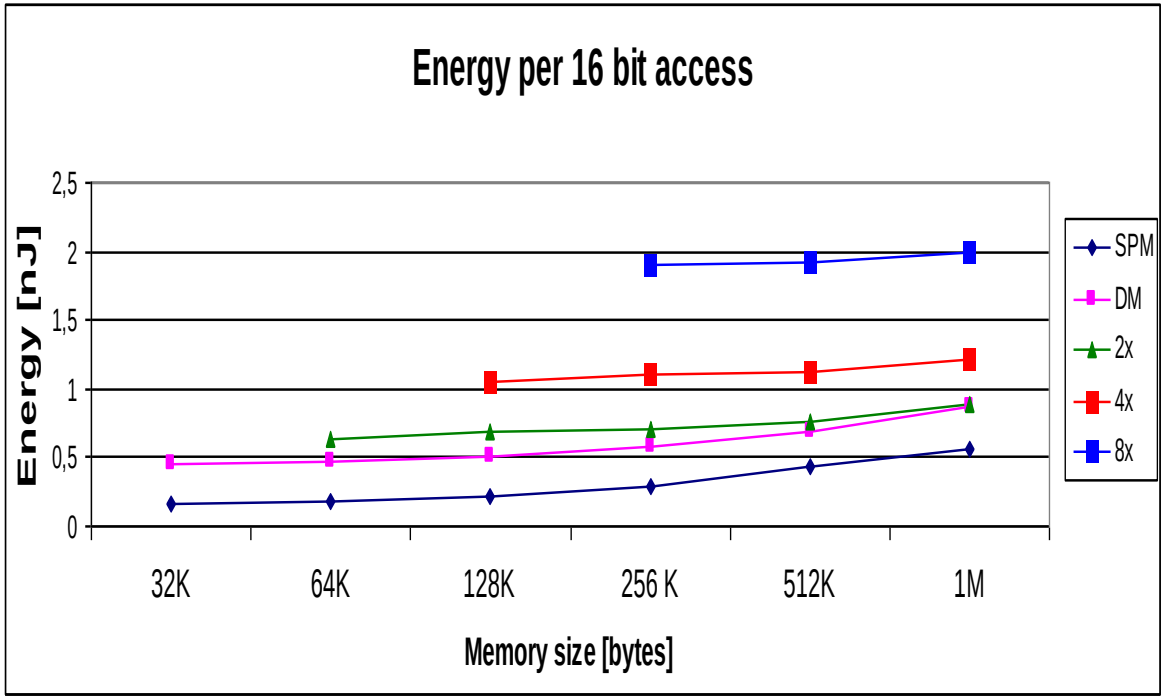


16 bit read; size in bytes;
65 nm for SRAM, 80 nm for DRAM

Source: Olivera Jovanovic, TU Dortmund, 2011

Why not just use a cache ?

- Energy consumed in tags, comparators and muxes is large



Cacti-based
16 banks
high performance
65 nm technology

O. Jovanovic, TU Dortmund, 2012

- Cache coherency (no stale data!) increasingly difficult to implement for a growing number of cores

Predictability and scratch-pad memories

*… **pre-run-time scheduling** is often the only practical means of providing predictability in a complex system.*

J. Xu, D. Parnas: On satisfying timing constraints in hard real-time systems, IEEE Trans. Soft. Engineering, 1993, p. 70–84

*… In essence, we must reinvent computer science. Fortunately, we have quite a bit of knowledge and experience to draw upon. Architecture techniques such as **software-managed caches** promise to deliver much of the benefit of memory hierarchy without the timing unpredictability.*

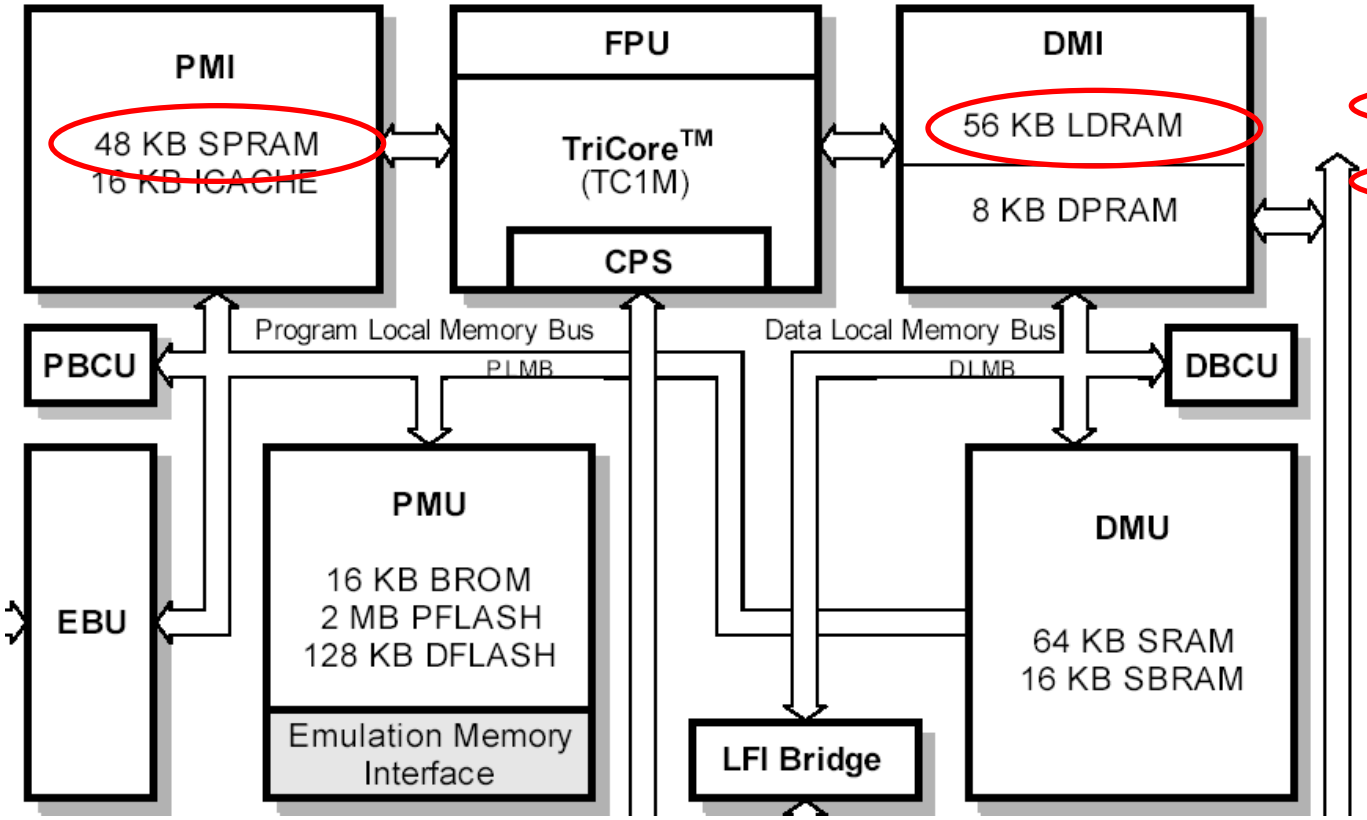
Edward Lee: Absolutely Positively on Time: What would it take?, IEEE Computer, 2005

Availability of SPMs ("Tightly Coupled Memories")

ARM CPU Core	TCM
Cortex-R4	Max. 8 MB
Cortex-R4(F)	Max. 8 MB
Cortex-R5	Max. 8 MB
Cortex R7	Max. 128 kB
ARM1136J(F)-S	Max. 64 kB
ARM1156T2(F)-S	Max. 25 kB
ARM1176JZ(F)-S	Max. 64 kB
ARM926EJ-S	Max. 1 MB
ARM946E-S	Max. 4 kB
ARM968E-S	Max. 4 MB

http://www.arm.com/products/processors/selector.php, Jan. 2014

Infineon TriCore

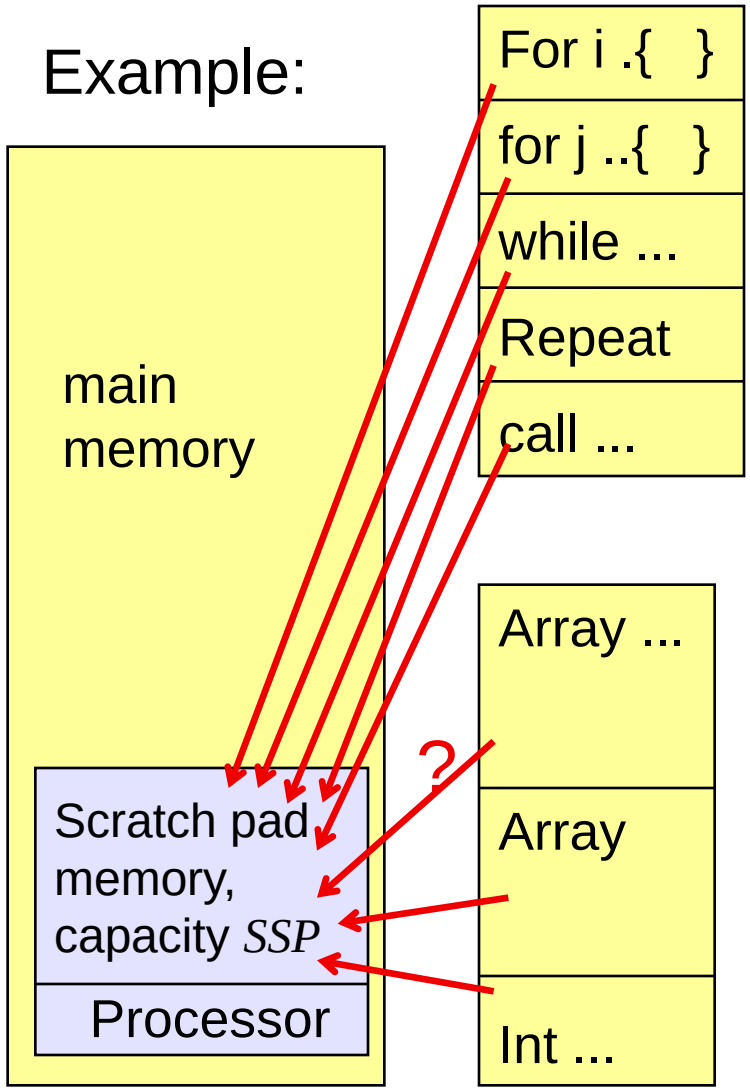


SPRAM:	Scratch-Pad RAM
ICACHE:	Instruction cache
LDRAM	Local data RAM
DPRAM:	Dual-port RAM
BROM:	Boot ROM
PFLASH:	Program Flash Mem
DFLASH:	Data Flash Memory
SBRAM:	Stand-by Data Mem
SRAM:	Data Memory
PRAM:	PCP Parameter Me
CMEM:	PCP Code Memory
PLMB:	Program Local Mem
DLMB:	Data Local Memory
RPB:	Remote Peripheral
SPB:	System Peripheral E
shaded:	only available in TC

Many more SPMs

- Early computers like IBM 360 model 25 and others
- Cyrix 6x86: 256 Byte SPM
- Sony PS1 and PS2
- M-core µcontr. (Freescale): 8-32 kB, on-chip, 1 cycle acc.
- NVIDIA Fermi GPU
- Merrimac supercomputer (U Stanford):
  768 registers, 8 k x 64 bits SPM
- Cyclops 64 (DoD, DoE, IBM):
  80 processors per chip, each with a 32 kB SPM [Wikipedia]
- Grape-DR (U. of Tokio), 256 x 72 bits local memory
- Many digital signal processors
- Processors with locked cache lines
- PhysX

Migration of data & instructions, global optimization model (TU Dortmund)



Which memory object (array, loop, etc.) to be stored in SPM?

Non-overlaying ("Static") allocation:

Gain g_k and size s_k for each object k. Maximise gain $G = \Sigma g_k$, respecting size of SPM $SSP \geq \Sigma\ s_k$.

Solution: knapsack algorithm.

Overlaying ("dynamic") allocation:

Moving objects back and forth

Pre-requisite: **Integer linear programming models**

Ingredients:

- Cost function
- Constraints

} Involving linear expressions of integer variables from a set X

Cost function $C = \sum_i a_i x_i$ with $a_i \in \mathbb{R}$, $x_i \in \mathbb{N}$ (1)

Constraints: $\forall j \in J$: $\sum_i b_{i,j} x_i \geq c_j$ with $b_{i,j}$, $c_{i,j} \in \mathbb{R}$ (2)

Def.: The problem of minimizing (1) subject to the constraints (2) is called an **integer linear programming (ILP) problem**.

If all x_i are constrained to be either 0 or 1, the ILP problem said to be a **0/1 integer linear programming problem**.

Pre-requisite: **Example**

$$C = 5x_1 + 6x_2 + 4x_3$$

$$x_1 + x_2 + x_3 \geq 2$$

$$x_1, x_2, x_3 \in \{0,1\}$$

x_1	x_2	x_3	C	
0	1	1	10	
1	0	1	9	← Optimal
1	1	0	11	
1	1	1	15	

ILP representation
- migrating functions and variables-

Symbols:

$S(var_k)$ = size of variable k

$n(var_k)$ = number of accesses to variable k

$e(var_k)$ = energy **saved** per variable access, if var_k is migrated

$E(var_k)$ = energy **saved** if variable var_k is migrated (= $e(var_k)\ n(var_k)$)

$x(var_k)$ = decision variable, =1 if variable k is migrated to SPM,
=0 otherwise

K = set of variables; similar for functions I

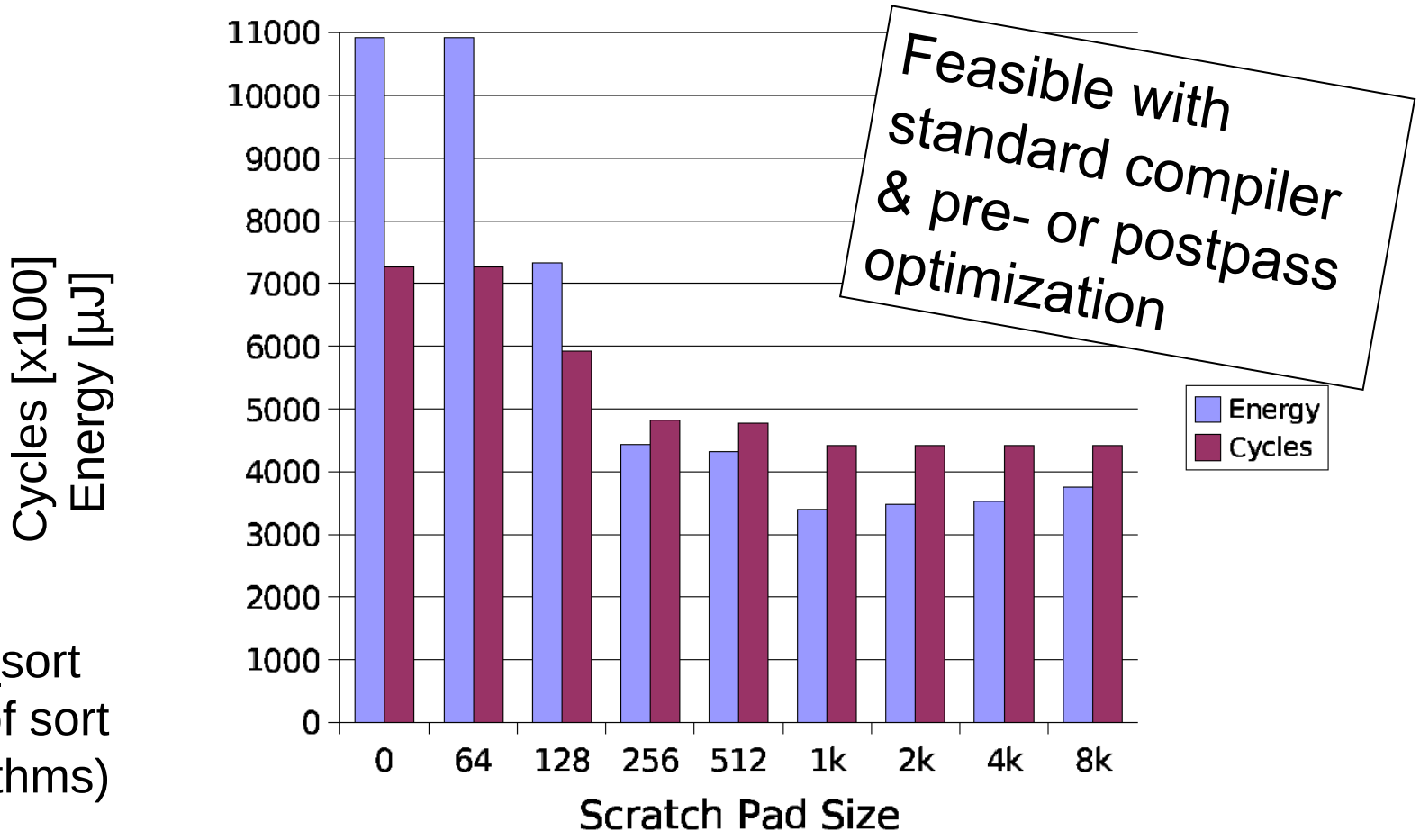
Integer programming formulation:

Maximize $\sum_{k \in K} x(var_k)\ E(var_k) + \sum_{i \in I} x(F_i)\ E(F_i)$

Subject to the constraint

$\sum_{k \in K} S(var_k)\ x(var_k) + \sum_{i \in I} S(F_i)\ x(F_i) \leq SSP$

Reduction in energy and average run-time



Multi_sort (mix of sort algorithms)

Measured processor / external memory energy + CACTI values for SPM (combined model)

Numbers will change with technology, algorithms remain unchanged.

Veröffentlichung mit recht vielen Zitaten

Scratchpad Memory : A Design Alternative for Cache On-chip memory in Embedded Systems

Rajeshwari Banakar, Stefan Steinke, Bo-Sik Lee, M. Balakrishnan, Peter Marwedel
banakar | mbala@cse.iitd.ernet.in
Indian Institute of Technology, Delhi 110 016
steinke | lee | marwedel@ls12.cs.uni-dortmund.de
University of Dortmund, Dept. of Computer Science
44221 Dortmund, Germany

ABSTRACT

In this paper we address the problem of on-chip memory selection for computationally intensive applications, by proposing scratch pad memory as an alternative to cache.

have on-chip scratch pad memories. In cache memory systems, the mapping of program elements is done during runtime, whereas in scratch pad memory systems this is done either by the user or automatically by the compiler using suitable algorithm.

Peter Marwedel, Rajeshwari Banakar (Delhi, March 2013)

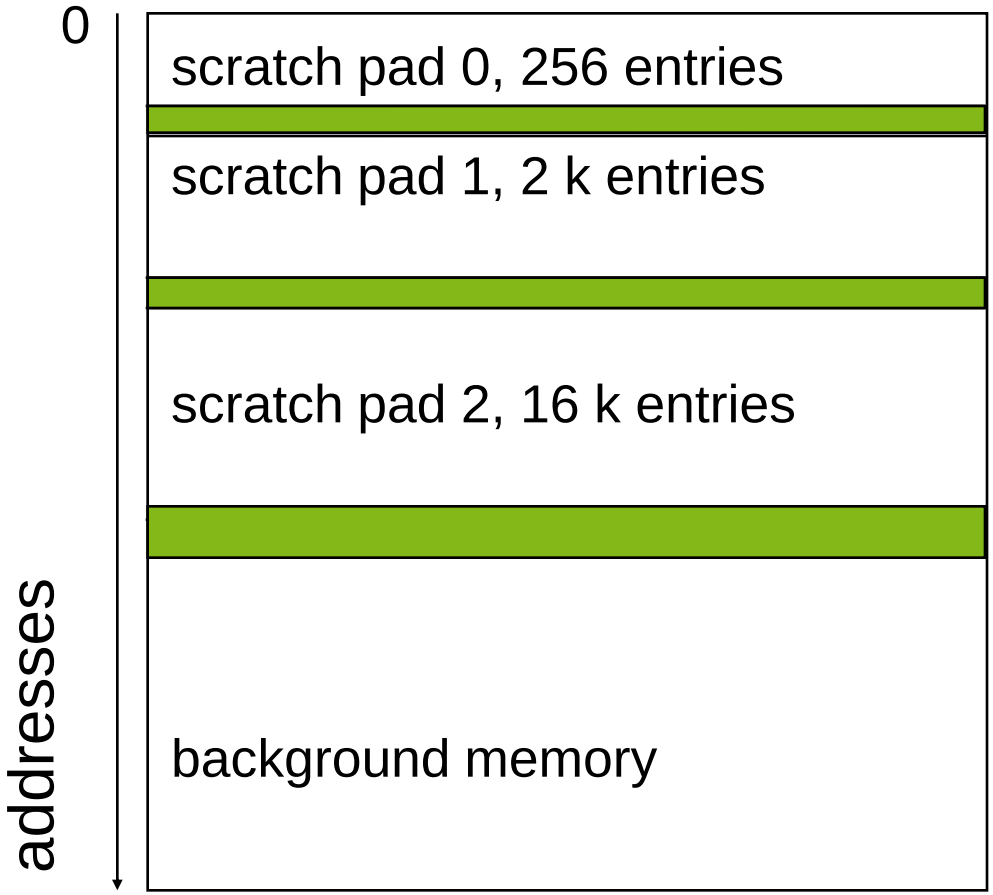
Scratchpad memory: design alternative for cache on-chip memory in embedded systems
R **Banakar**, S Steinke, BS Lee, M Balakrishnan... - Proceedings of the ..., 2002 - dl.acm.org
Abstract In this paper we address the problem of on-chip memory selection for computationally intensive applications, by proposing scratch pad memory as an alternative to cache. Area and energy for different scratch pad and cache sizes are computed using ...
Cited by 535 Related articles All 14 versions Import into BibTeX Save More

Partitioning

Small is beautiful:

One small SPM is beautiful (☺).

May be, several smaller SPMs are even more beautiful?



Considered partitions

# of partitions	number of partitions of size:						
	4k	2k	1k	512	256	128	64
7	0	1	1	1	1	1	2
6	0	1	1	1	1	2	0
5	0	1	1	1	2	0	0
4	0	1	1	2	0	0	0
3	0	1	2	0	0	0	0
2	0	2	0	0	0	0	0
1	1	0	0	0	0	0	0

Example of considered memory partitions for a total capacity of 4096 bytes

Optimization for multiple scratch pads

Minimize $C = \sum_j e_j \cdot \sum_i x_{j,i} \cdot n_i$

With e_j: energy ***required*** per access to memory j,
and $x_{j,i}$= 1 if object i is mapped to memory j, =0 otherwise,
and n_i: number of accesses to memory object i,
subject to the constraints:

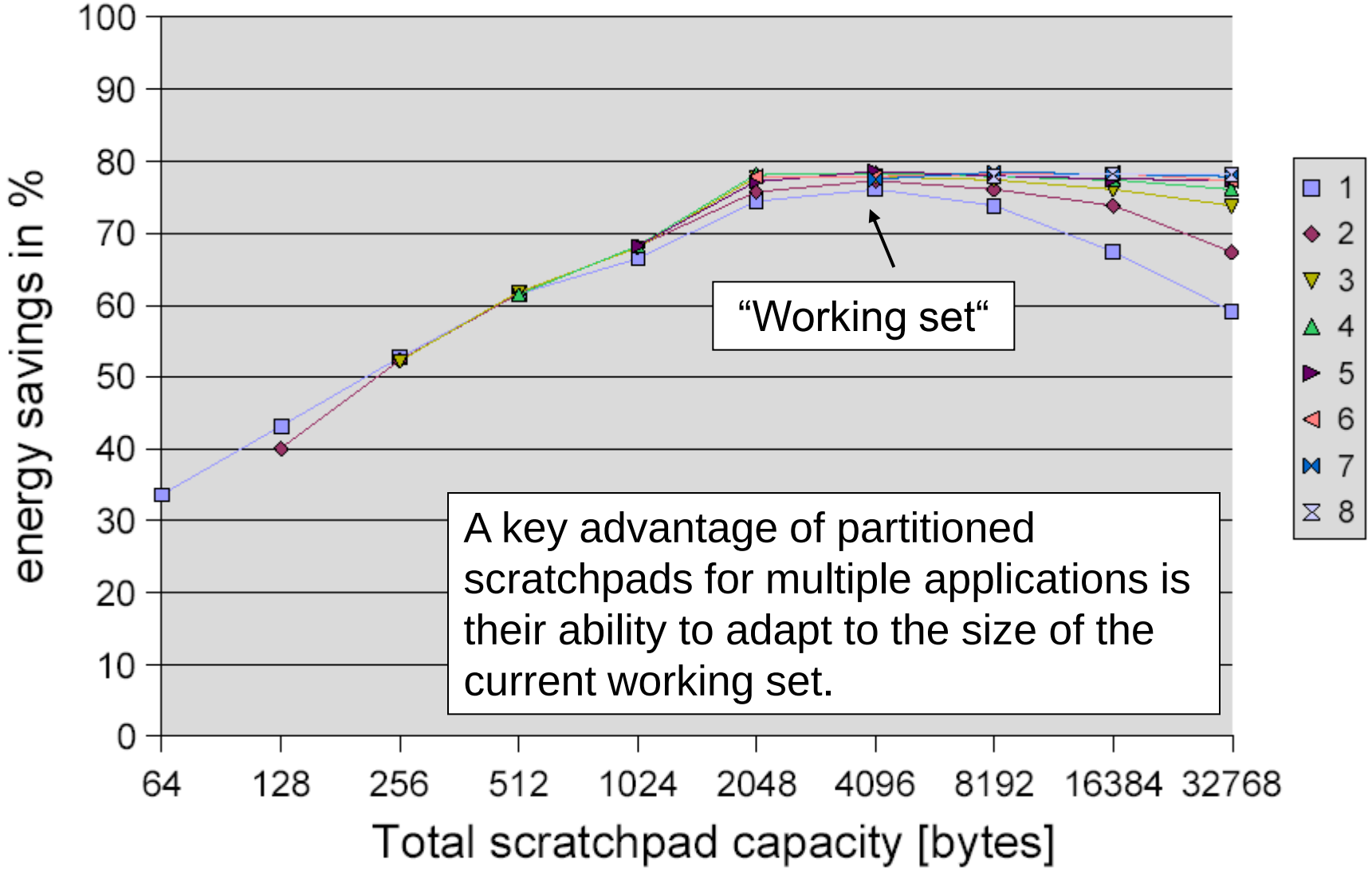
$$\forall j : \sum_i x_{j,i} \cdot S_i \leq SSP_j$$

$$\forall i : \sum_j x_{j,i} = 1$$

With S_i: size of memory object i,
SSP_j: size of memory j.

Main memory included as a special case of j

Results for parts of GSM coder/decoder



How much better can we get?

$$improvement = \frac{1}{(1-P) + \frac{P}{S}}$$

(Amdahl's law)
where

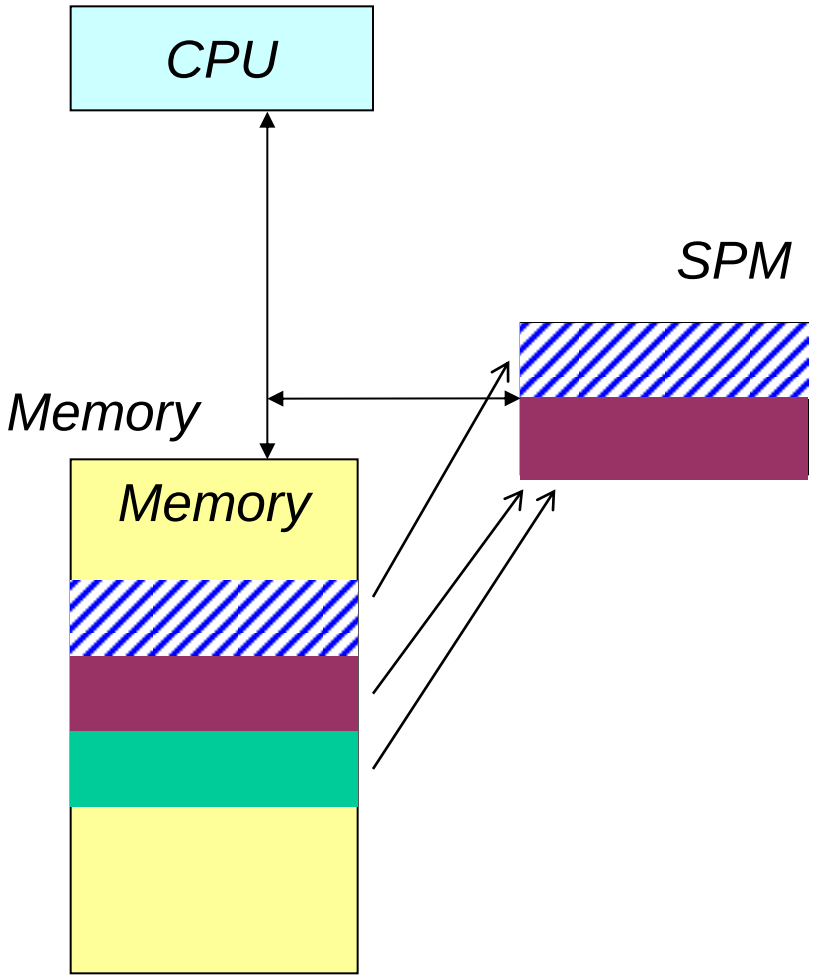
- P: fraction of memory references replaced by faster/more energy efficient memory

and

- S: speed/energy improvement

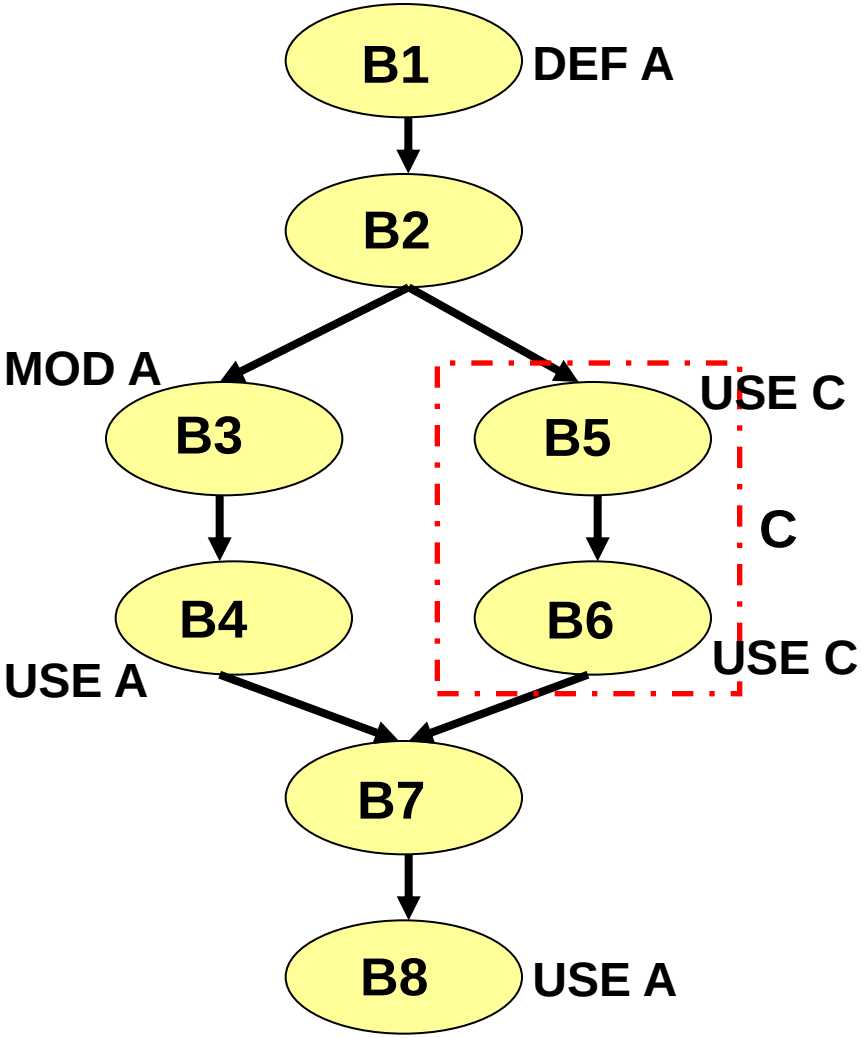
Important not to have too many "untouchable" references (1-P), otherwise even $S \to \infty$ does not help

Non-overlaying allocation problematic for multiple hot spots ☞Overlaying allocation



- Effectively results in a kind of **compiler-controlled overlays** for SPM
- Address assignment within SPM required

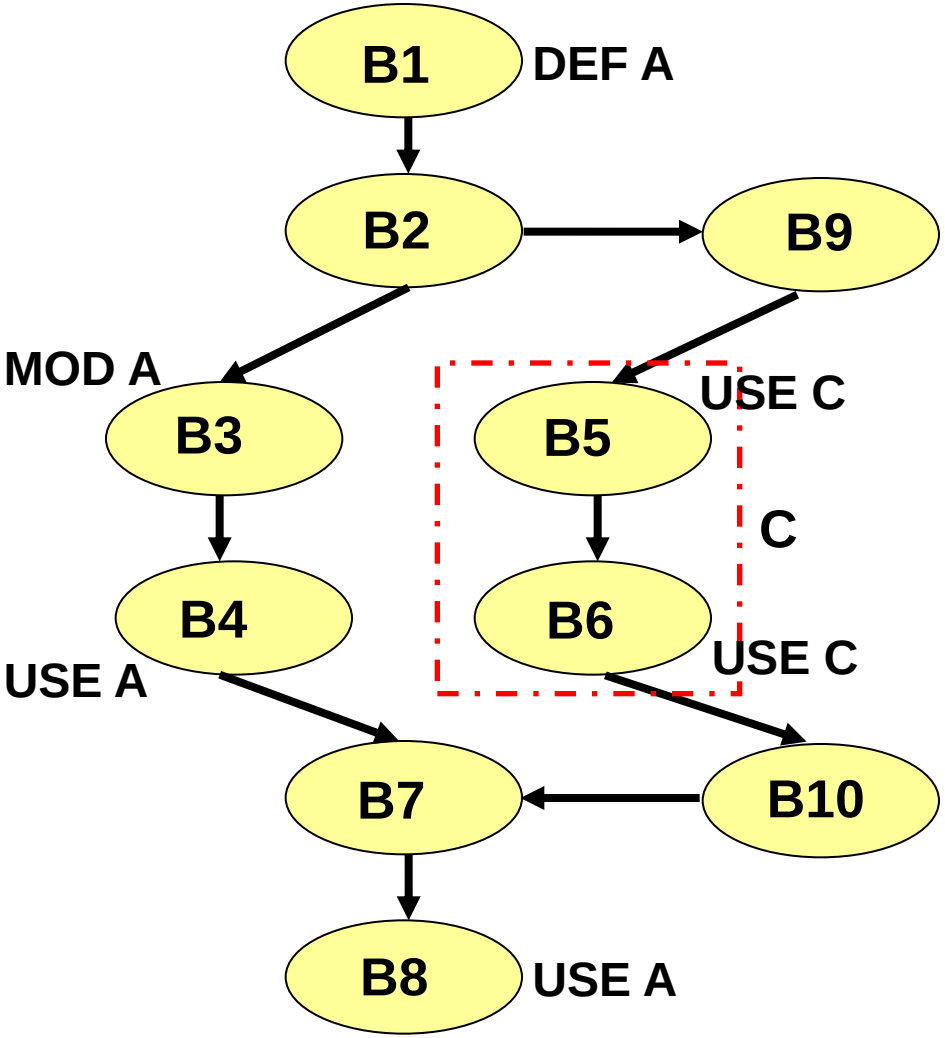
Overlaying allocation by Verma et al. (1)



Based on control flow graph.

[M.Verma, P.Marwedel: Dynamic Overlay of Scratchpad Memory for Energy Minimization, *ISSS,* 2004]

Overlaying allocation by Verma et al. (2)



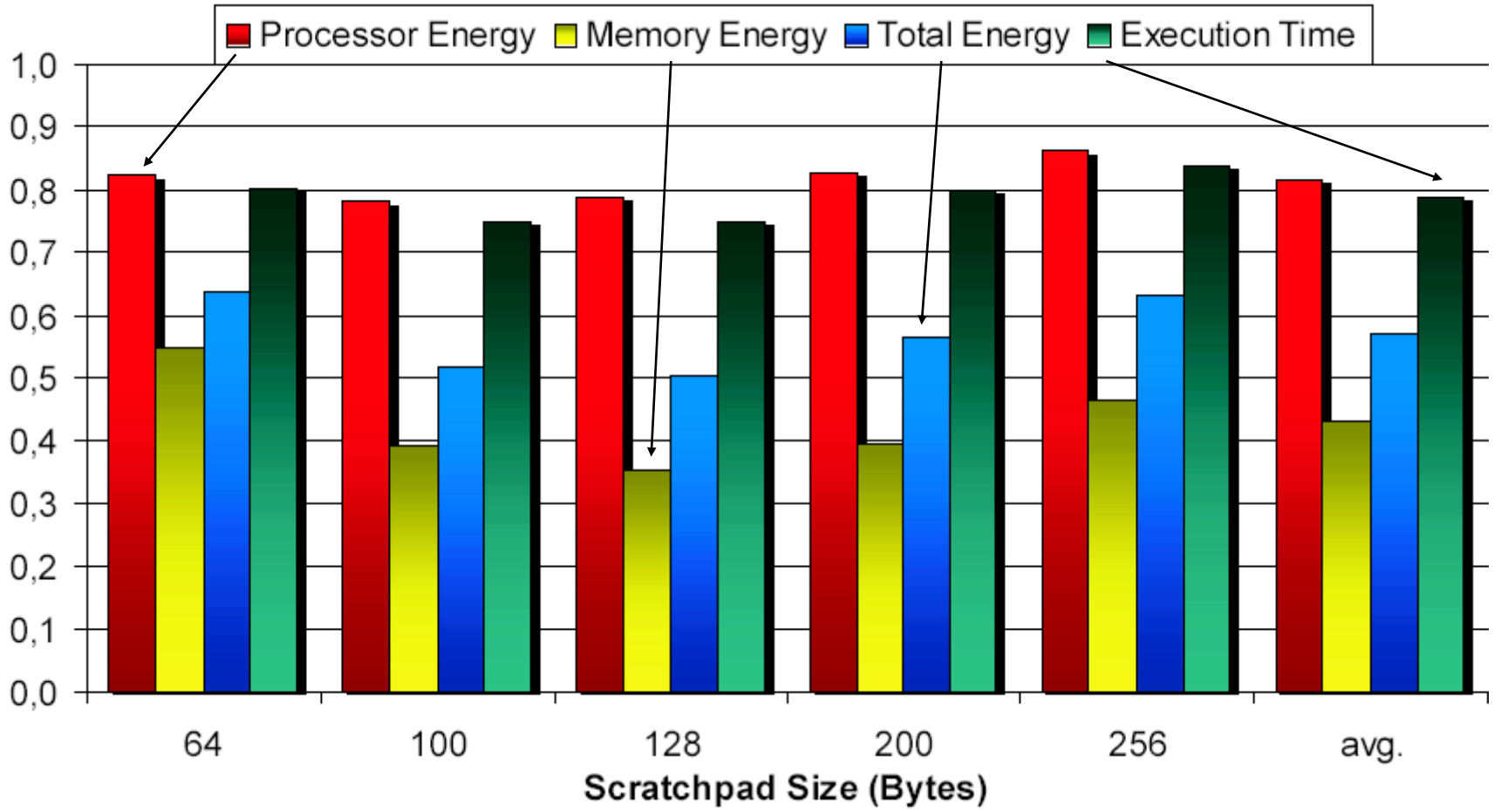

```
SPILL_STORE(A);
SPILL_LOAD(C);
```

Global set of ILP equations reflects cost/benefit relations of potential copy points

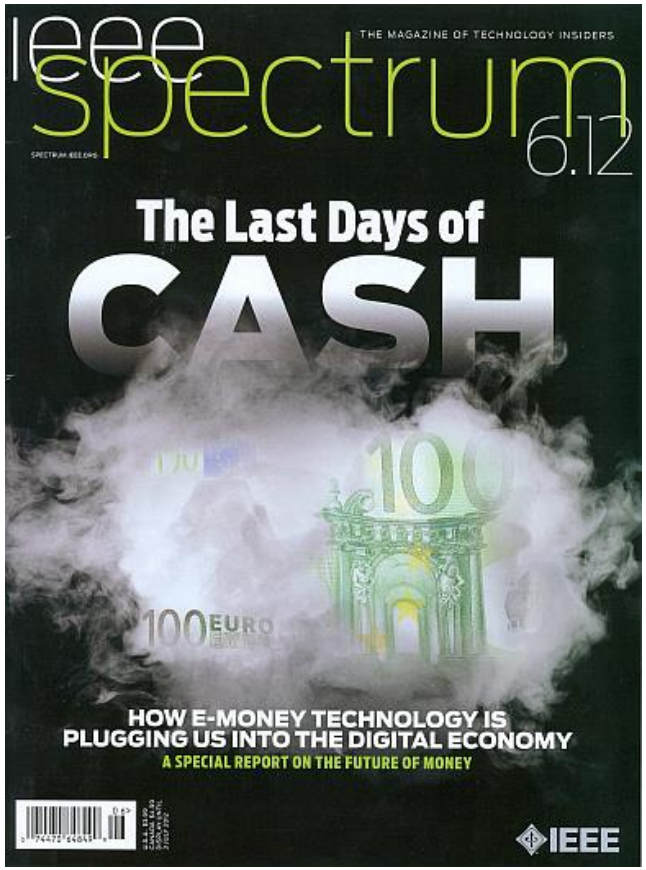
```
SPILL_LOAD(A);
```

Code handled like data

Runtime/energy reduction with respect to non-overlaying (“static”) allocation



Less seriously …



☞ Some people got already completely rid of cache

Speicherhierarchie: Scratch Pad- und Flash-Speicher

Peter Marwedel
Informatik 12

Funktionen pro Chip

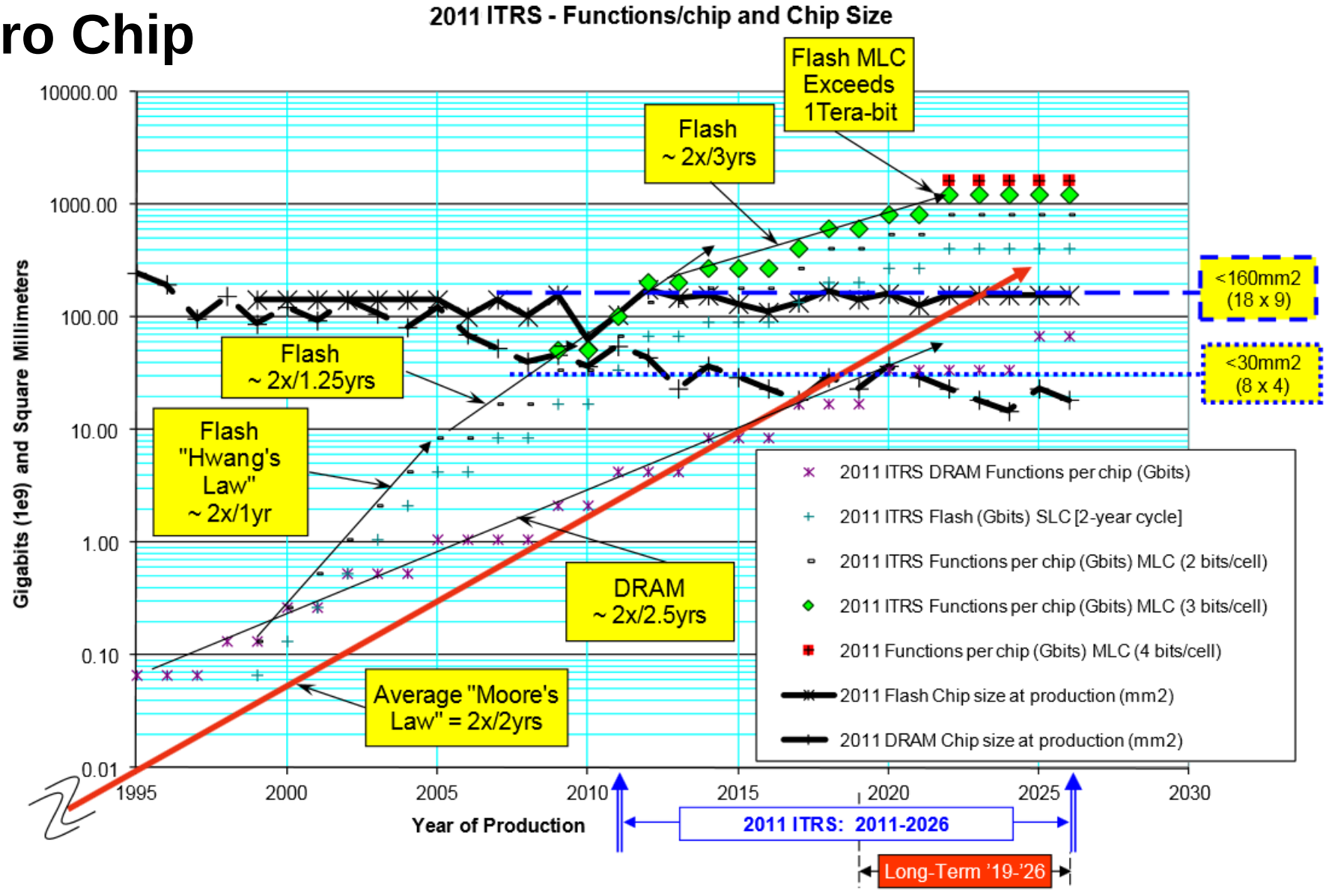
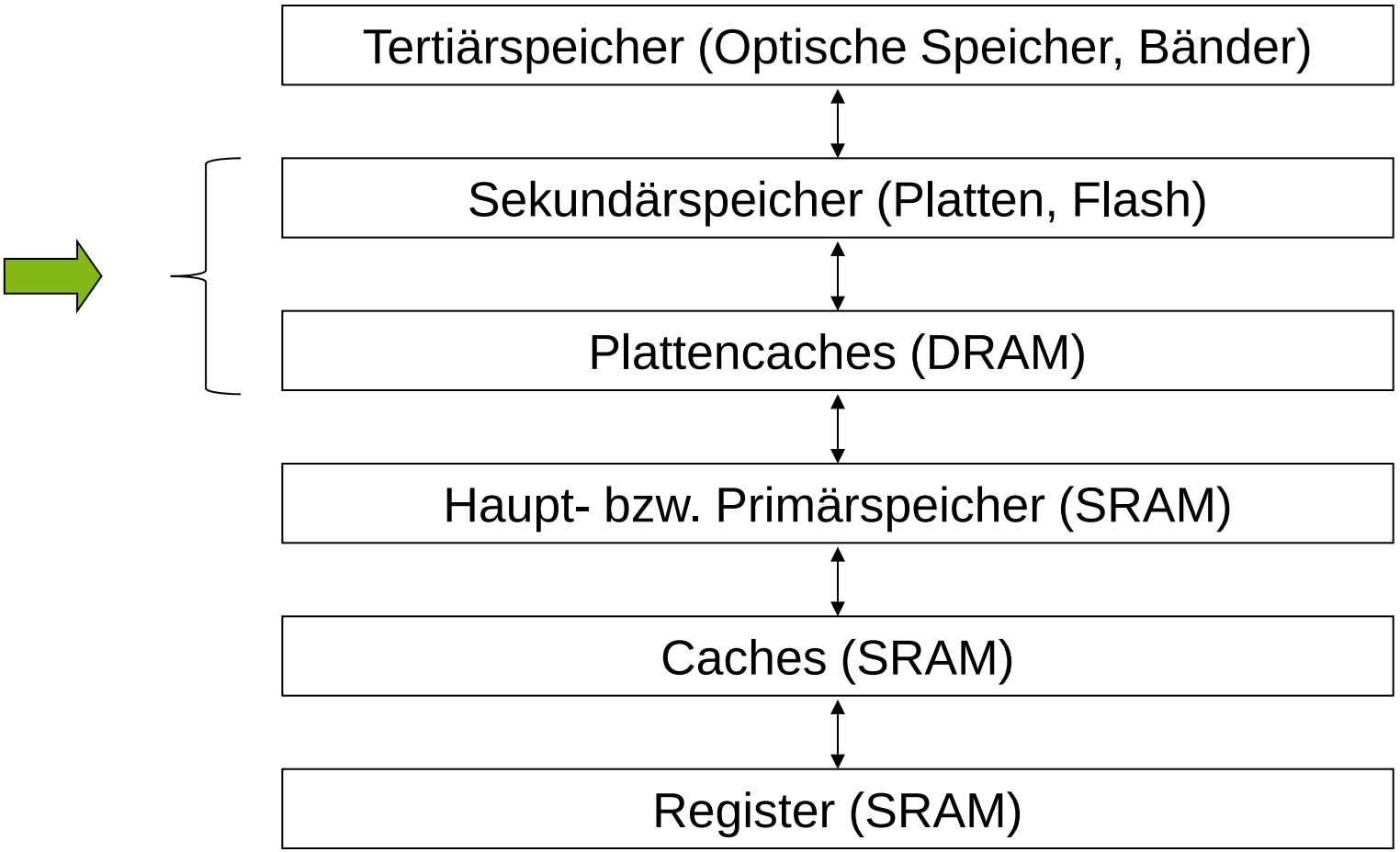


Figure ORTC7 2011 ITRS Product Technology Trends: Memory Product Functions/Chip and Industry Average "Moore's Law" and Chip Size Trends

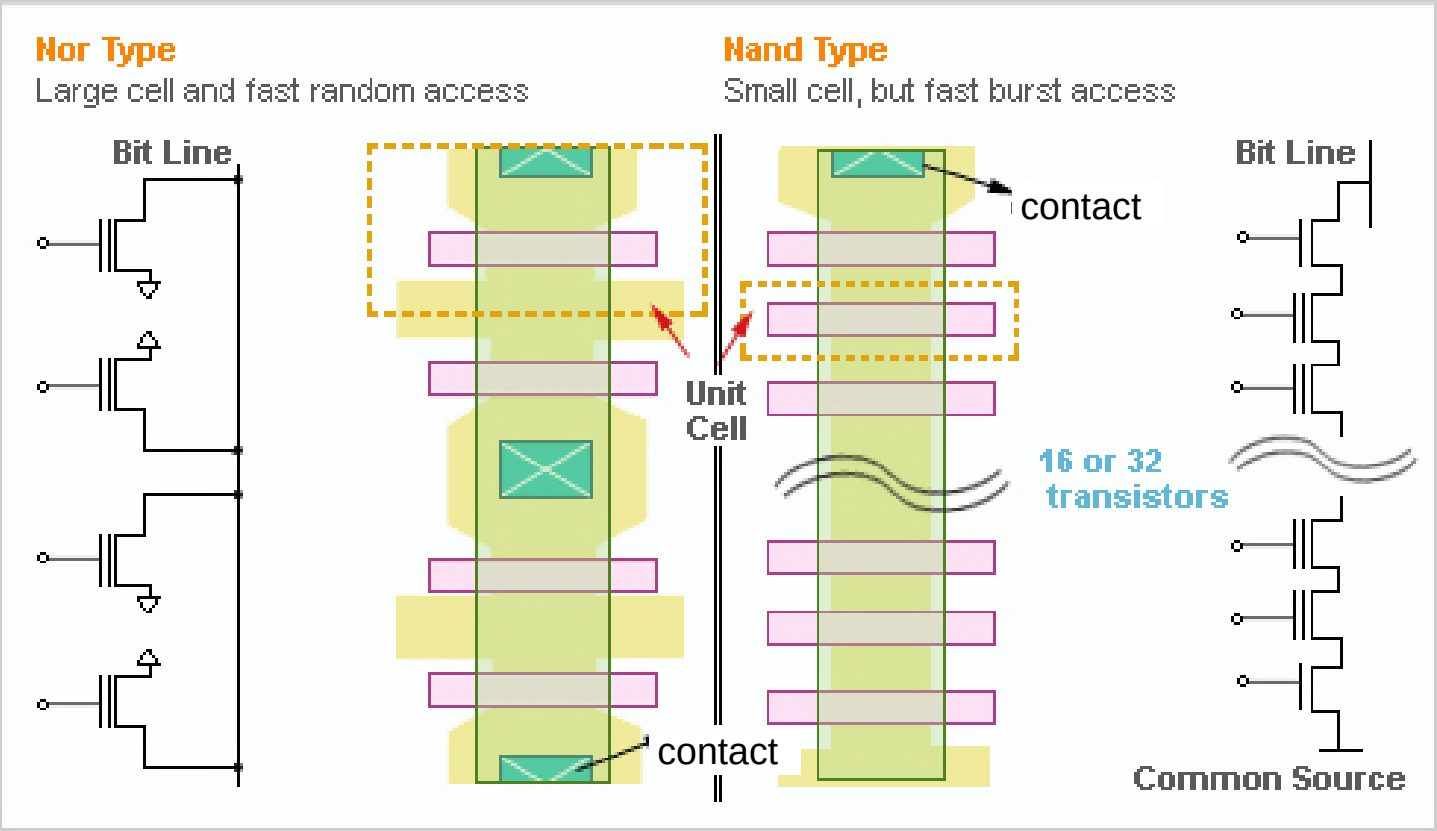
[ITRS 2011

Mögliche Stufen der Speicherhierarchie
und derzeit eingesetzte Technologien



NOR- und NAND-Flash

NOR: 1 Transistor zwischen Bitleitung und Masse
NAND: >1 Transistor zwischen Bitleitung und Masse

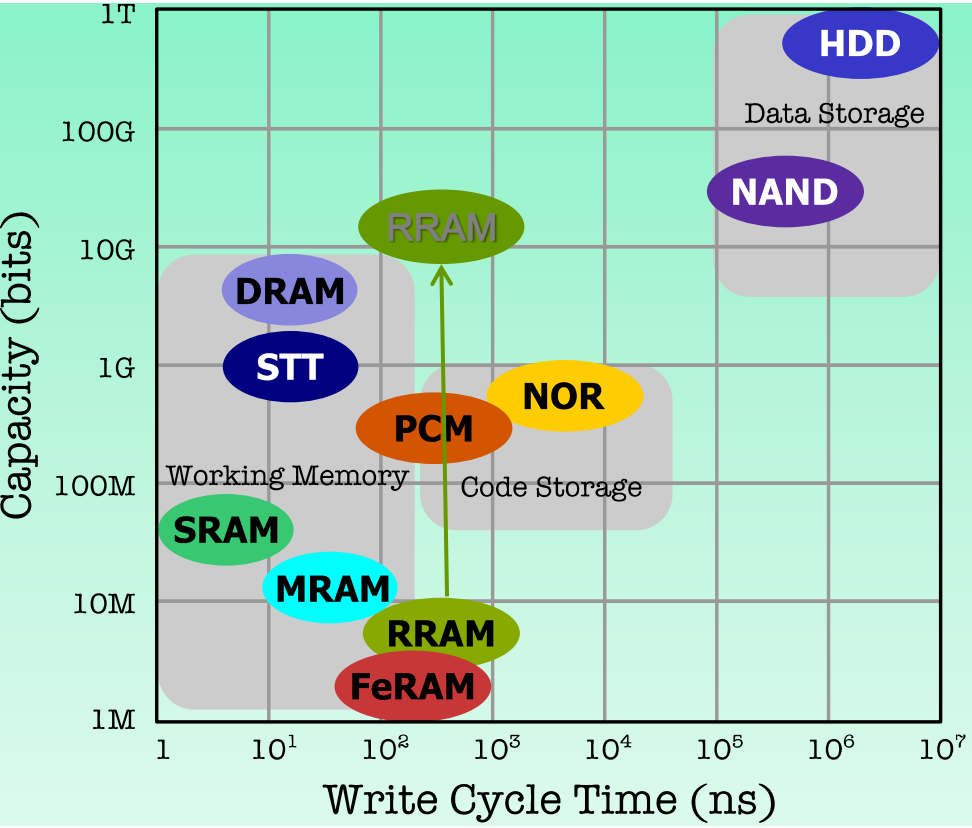


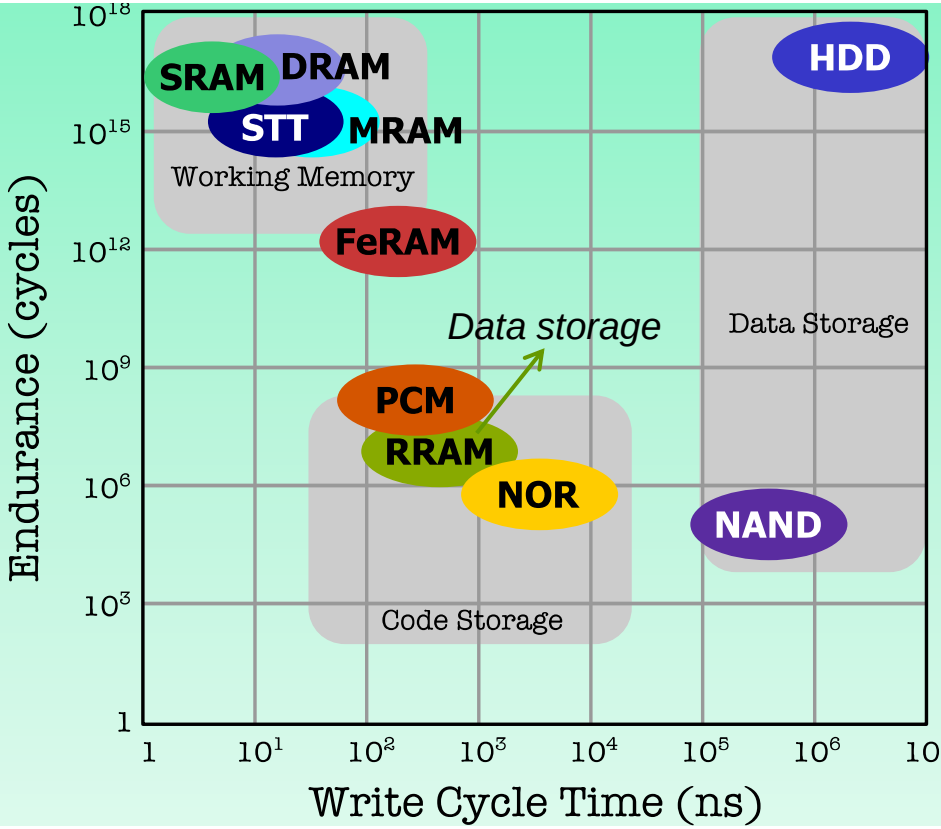
was at [www.samsung.com/Products/ Semicon-ductor/ Flash/FlashNews/ FlashStructure. htm] (2007)

Eigenschaften von NOR- und NAND-Flash-Speichern

Type/Eigenschaft	NOR	NAND
Wahlfreier Zugriff	Ja ☺	Nein ☹
Block löschen	Langsam ☹	Schnell ☺
Zellgröße	Groß ☹	Klein ☺
Zuverlässigkeit	Größer ☺	Kleiner ☹
Direktes Ausführen	Ja ☺	Nein ☹
Anwendungen	Codespeicherung, *boot flash, set top box*	Datenspeicher, USB *Sticks*, Speicherkarten

Landscape of Future Memories





SOURCE: STT-RAM: The Coming Revolution in Memory (11/6/2010) Future Fab Intl. Issue 35 By Alexander Driskill-Smith, Grandis

Key parameters:
- Write speed
- Endurance

Size, cost and maturity (yield) matters:
- $4F^2$ will be the size of future cells
- Material challenge

DRAM and NAND Flash still dominating!

© Christian Weis, 3D-Memories (DRAMs) - A Solution to the Memory Wall? – EWME 2014, Tallinn

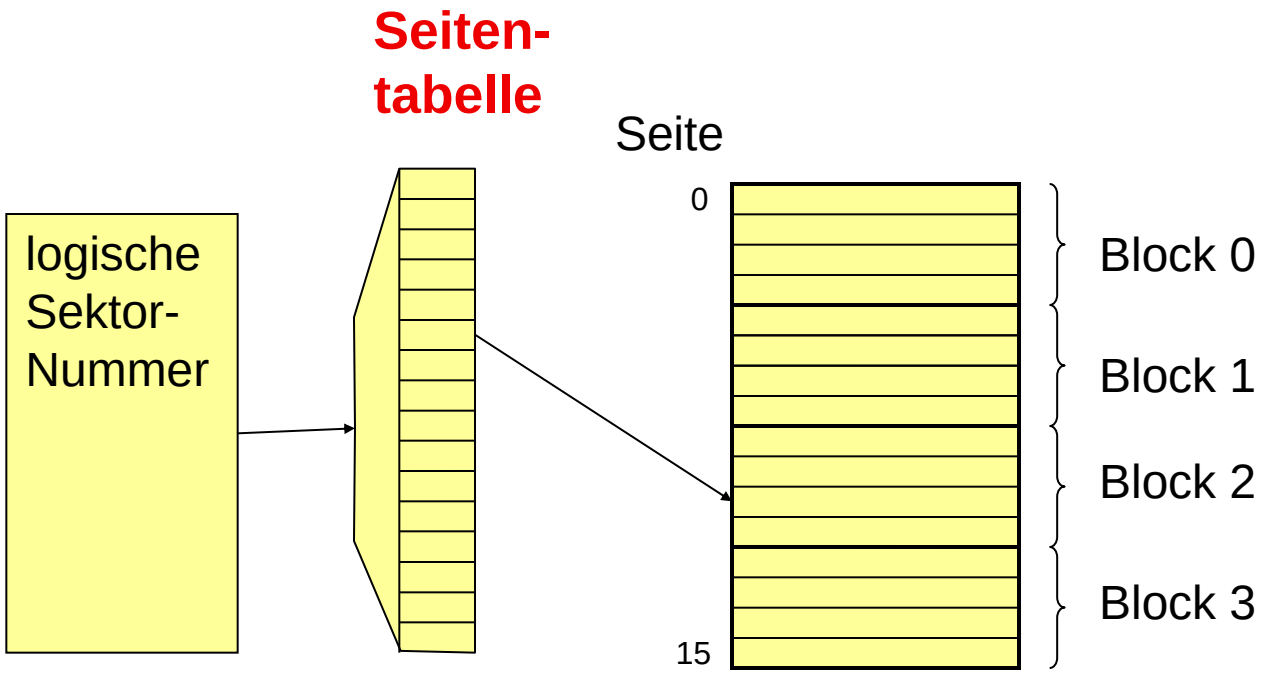
Charakteristische Eigenschaften von NAND Flash Speicher

Speicher aufgeteilt in Blöcke (typ. 16-256 KB),
Blöcke unterteilt in Seiten (typ. 0.5-5 KB).
Schreib-/Lesevorgänge jeweils auf Seiten

	1 Bit/Zelle (SLC)	>1 Bit/Zelle (MLC)
Lesen (Seite)	25 µs	≫ 25 µs
Schreiben (Seite)	300 µs	≫ 300 µs
Löschen (Block)	2 ms	1.5 ms

J. Lee, S. Kim, H. Kwin, C. Hyun, S, Ahn, J. Choi, D. Lee, S.Noh: Block Recycling Schemes and Their Cost-based Optimization in NAND Flash Memory Based Storage System, EMSOFT'07, Sept. 2007

Seiten-/bzw. Sektorabbildung mit *Flash transaction layer* (FTL)

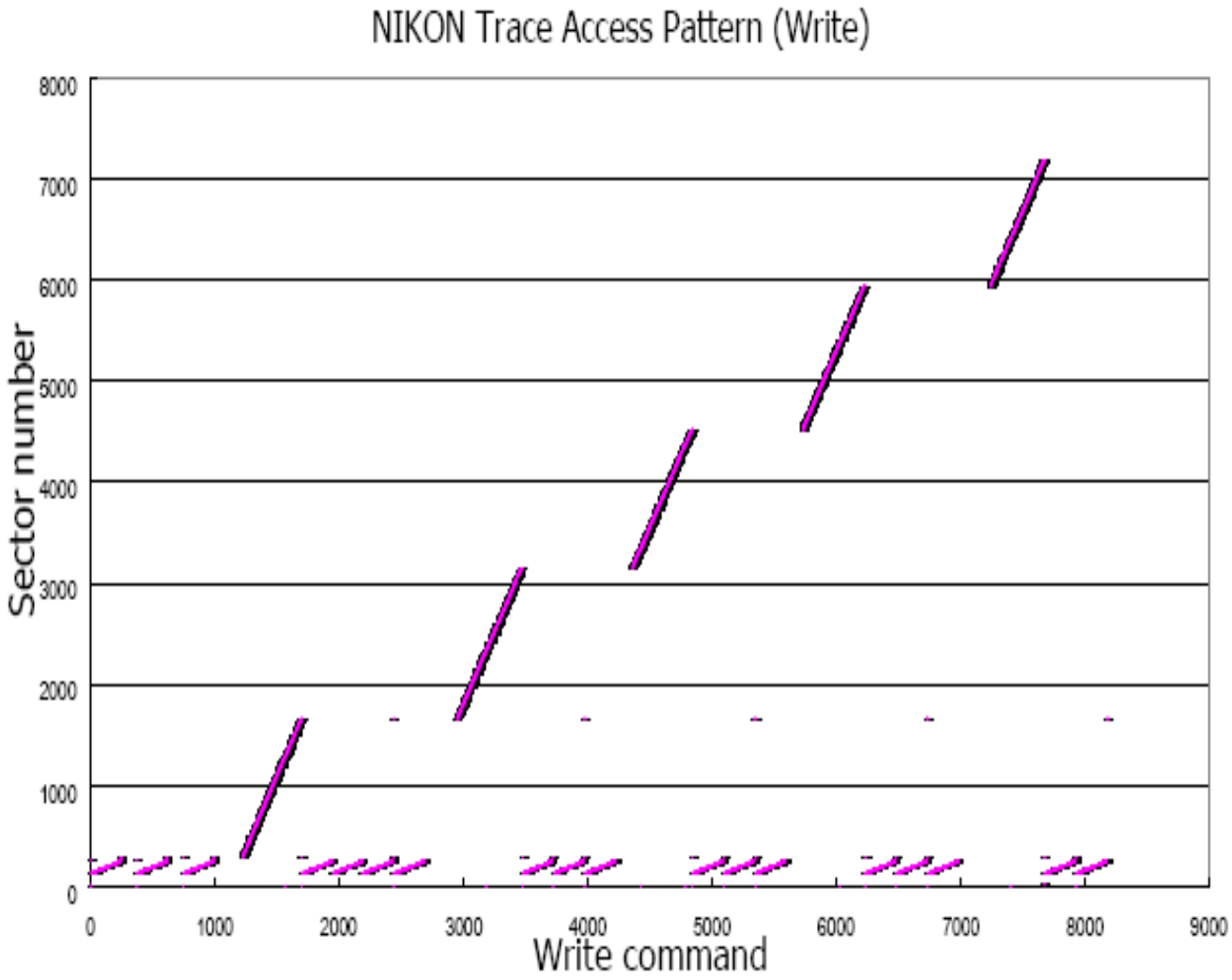


Invertierte Seitentabelle im Flashspeicher gespeichert (Extra Bits); “Normale Seitentabelle” während der Initialisierung erzeugt; Seitentabelle kann sehr groß werden; Wird in kleinen NOR Flash-Speichern benutzt.

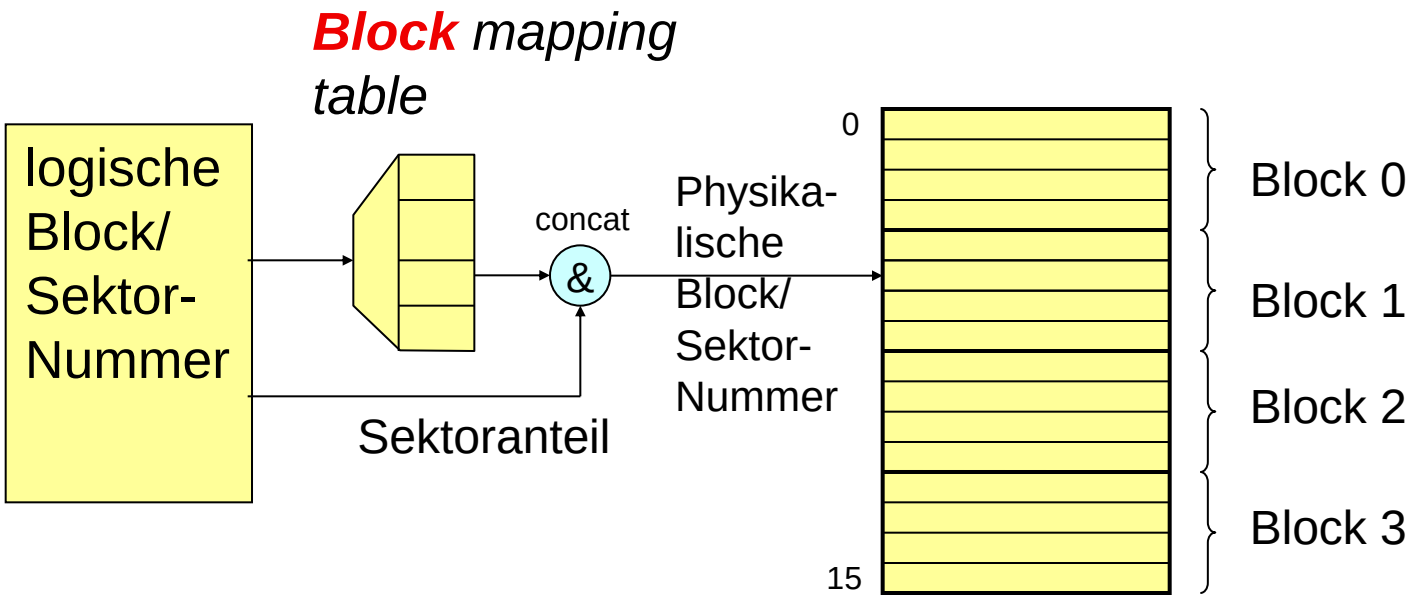
Sektor ≈ Seite
+ Extra Bits

Ausnutzung von Regularität

Häufig lange Sequenzen von sequentiellen Schreib-vorgängen



Block mapping flash transaction layer (FTL)

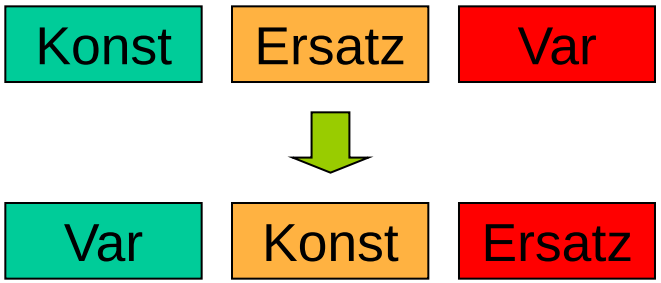


- Abbildungstabellen kleiner als bei Seiten-basierten FTLs
- ☞ In großen NAND Flash-Speichern benutzt
- ☞ Einfache Realisierung,
- Wiederholtes Schreiben erfordert Kopieren auf einen neuen Block
- Schlechte *Performance* bei wiederholtem und zufälligem Schreiben
- Hybride Verfahren

Ausgleich der Abnutzung (*wear levelling*)

Beispiel (Lofgren et al., 2000, 2003):

- Jede *erase unit* (Block) besitzt Löschzähler
- 1 Block wird als Ersatz vorgehalten
- Wenn ein häufig genutzter Block frei wird, wird der Zähler gegen den des am wenigsten benutzten Blocks verglichen. Wenn der Unterschied groß ist:
  - Inhalt wenig genutzten Blocks (≈ Konstanten) → Ersatz
  - Inhalt häufig genutzten Blocks → am wenigsten genutzter Block
  - Häufig genutzter Block wird zum Ersatzblock



Source: Gal, Toledo, *ACM Computing Surveys*, June 2005

Flash memory as main memory

Approach published (Wu, Zwaenepoel, 1994):

- Uses MMU
- RAM + Flash mapped to memory map
- Reads from Flash read single words from Flash
- Writes copy block of data into RAM,
  all updates done in RAM
- If the RAM is full, a block is copied back to Flash
- Crucial issue: Speed of writes.
  Proposal based on wide bus between Flash and RAM, so that writes are sufficiently fast
  ☞ Larger erase units, increased wear-out feasible.

M. Wu, W. Zwaenepoel: eNVy: A nonvolatile, main memory storage system. In *Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems*. 1994, p. 86–97.

Flash-spezifische Dateisysteme

- Zwei Ebenen können ineffizient sein:
  - FTL bildet Magnetplatte nach
  - Standard-Dateisystem basiert auf Magnetplatten

  Beispiel: Gelöschte Sektoren nicht markiert
  ☞ nicht wieder verwendet
- Log-strukturierte Dateisysteme fügen nur neue Informationen zu
  - Für Magnetplatten
    - Schnelle Schreibvorgänge
    - Langsames Lesen (Kopfbewegungen für verteilte Daten)
  - Ideal für Flash-basiertes Dateisystem:
    - Schreibvorgänge in leere Sektoren
    - Lesen nicht langsam, da keine Köpfe bewegt werden

☞ Spezifische log-basierte *Flash*-Dateisysteme
- JFFS2 (NOR)
- YAFFS (NAND)

Source: Gal, Toledo, *ACM Computing Surveys*, June 2005

Vergleich *Harddisc/Flash*-Speicher (2011)

	Flash	HDD
Zugriffszeit (random) [ms]	~0.1	5-10
Kosten [$/GB]	1,2-2 (Fixanteil gering)	0,05-0,1, Fixanteil !
Kapazität [GB]	Typ. < 120	1000-3000
Leistungsaufnahme [W]	Ca. 1/3-1/2 der HDD-Werte	Typ. 12-18, laptops: ~2
Defragmentierung	unwichtig	Zu beachten
Zeitverhalten	Relativ deterministisch	Von Kopfbewegung abhängig
Anzahl der Schreibvorgänge	begrenzt	unbegrenzt
Verschlüsselung	Überschreiben unver-schlüsselter Info schwierig	Überschreiben einfach
Mechanische Empfindlichkeit	Gering	Stoßempfindlich
Wiederbenutzung von Blöcken	Erfordert Extra-Löschen	Überschreiben

What Is Flash Express?

- **Also referred to as Storage Class Memory (SCM)**
- **Flash Express is internal storage implemented via NAND Flash SSDs (Solid State Drives) mounted in PCIe Flash Express feature cards**
  - Plugs into PCIe I/O drawers in pairs
  - Data security provided on the feature cards
  - A pair provides 1.6 TB of storage
  - A maximum of 4 pairs are supported in a system
- **Internal Flash is accessed using the new System z architected EADM (Extended Asynchronous Data Mover) Facility**
  - An extension of the ADM architecture used in the past with expanded storage
  - Access is initiated with a Start Subchannel instruction
  - Subchannels used were previously reserved
  - Definition in IOCDS is not required
- **The main application of internal Flash in zEC12 is paging store for z/OS**
  - Where it provides advantages in resiliency and speed
  - With pageable large pages being introduced in tandem for exceptional performance

z/OS FLASH Use Cases

- **Paging**
  - z/OS paging subsystem will work with mix of internal Flash and External Disk
    - Self Tuning based on measured performance
    - Improved Paging Performance, Simplified Configuration
  - Begin Paging 1 MB Large Pages only to Flash
    - Exploit Flash's random I/O read rate to gain CPU performance by enabling additional use of Large Pages. Currently large pages are not pagable.
  - Begin Speculative Page-In of 4K Pages
    - Exploit Flash's random I/O read rate to get Improved Resilience over Disruptions.
    - Market Open, Workload Failover,

Re-writing algorithms for memory hierarchies

Analysis of algorithm complexity mostly using the *RAM* (random access machine; constant memory access times) model outdated

☞ take memory hierarchies explicitly into account.

Example:

- Usually, divide-&-conquer algorithms are good.
- "Cache"-oblivious algorithms (are good for any size of the faster memory and any block size). Assuming
  - Optimal replacement (Belady's algorithm)
  - 2 Memory levels considered (there can be more)
  - Full associativity
  - Automatic replacement

Unlikely to be ever automatic

[Piyush Kumar: Cache Oblivious Algorithms, in: U. Meyer et al. (eds.): Algorithms for Memory Hierarchies, *Lecture Notes in Computer Science, Volume 2625*, 2003, pp. 193-212]

[Naila Rahman: Algorithms for Hardware Caches and TLB, in: U. Meyer et al. (eds.): Algorithms for Memory Hierarchies, *Lecture Notes in Computer Science, Volume 2625*, 2003, pp. 171-192]

Zusammenfassung

Speicherhierarchie

- Scratchpadspeicher SPM *(„Software managed caches“)*
  - Schnell, energieeffizient, *timing predictable,*
  - Populär wg. Aufwands für *Cache*-Kohärenz in Multiprozessor-Syst.
  - Statische, nicht-überlagernde Allokation (Knappsack, ILP)
  - Überlagernde, dynamische *compile-time* Allokation (ILP)
  - *Run-time allocation* (im Betriebssystem)
- *Flash*-Speicher erfordern Anpassung an Eigenheiten
  - Ausgleich der Abnutzung
  - In der Regel Abbildung logische→reale Blockadressen (FTL/MMU)
  - Nur eingeschränkt als Hauptspeicher geeignet
  - Als Sekundärspeicher am besten mit speziellem Dateisystem!
- Große Datenmengen ☞ „Sekundärspeicher“ sind für die *Performance* die entscheidenden Komponenten