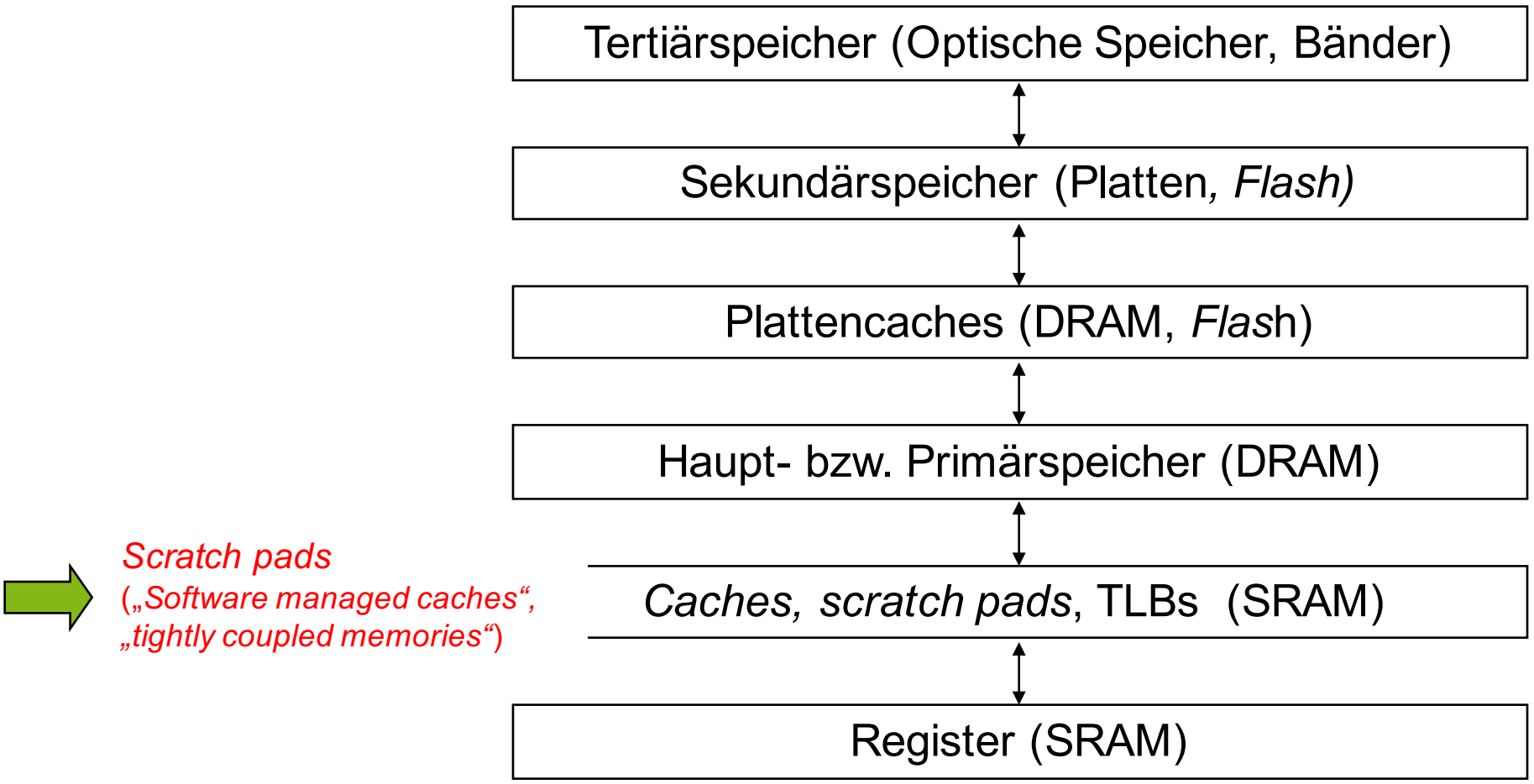
Rechnerarchitektur (RA)

Sommersemester 2020

Scratchpad Speicher

Jian-Jia Chen
Informatik 12
Jian-jia.chen@tu-..
http://ls12-www.cs.tu-dortmund.de/daes/
Tel.: 0231 755 6078

Mögliche Stufen der Speicherhierarchie
und derzeit eingesetzte Technologien



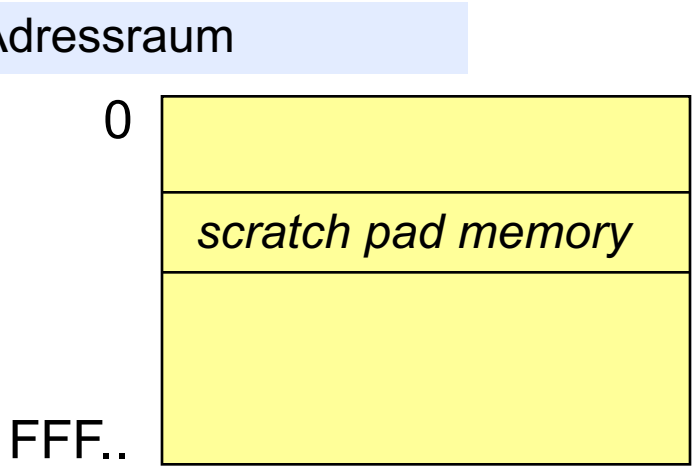
Scratch pad

Scratch pad?

© G. Marwedel, 2014

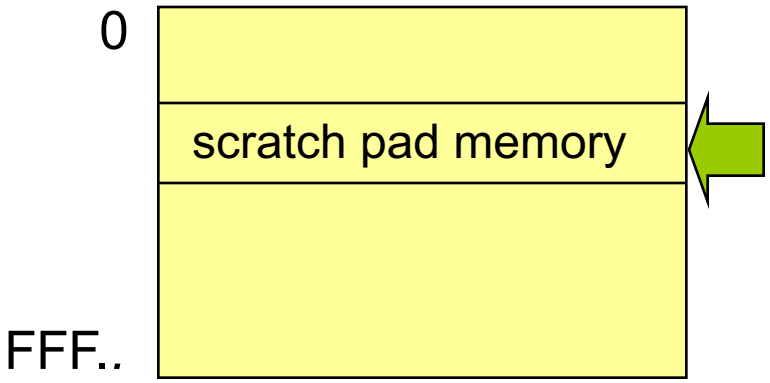
Scratch pad?

Scratch pad memory (SPM)!
SPMs sind kleine, physikalisch separate Speicher, die in den Adressraum abgebildet werden



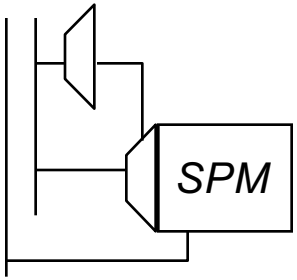
Scratch pad memories (SPM): schnell, energieeffizient, zeitgenau

Adressraum



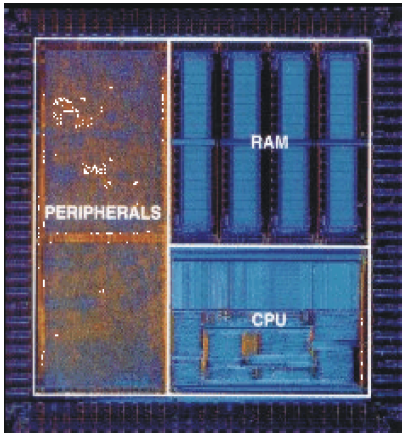
klein; ohne Tag

Man braucht nur einen Adressdekoder um die gültige Adresse vergleichen (einfach!)



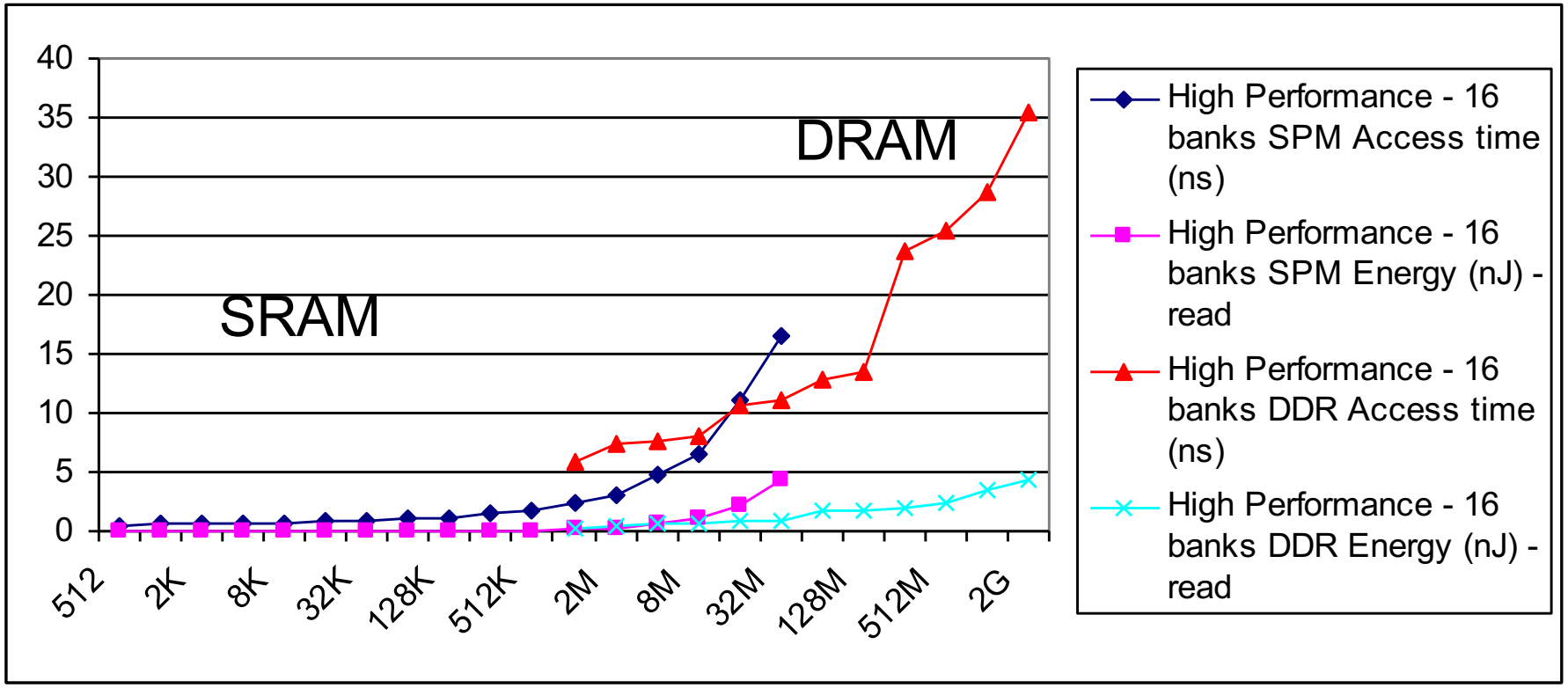
Beispiel

ARM7TDMI Prozessoren, bekannt für geringe Leistungsaufnahme



Energieverbrach des Speichers

"Small is beautiful": SRAM & DRAM (DDR2) [von CACTI]

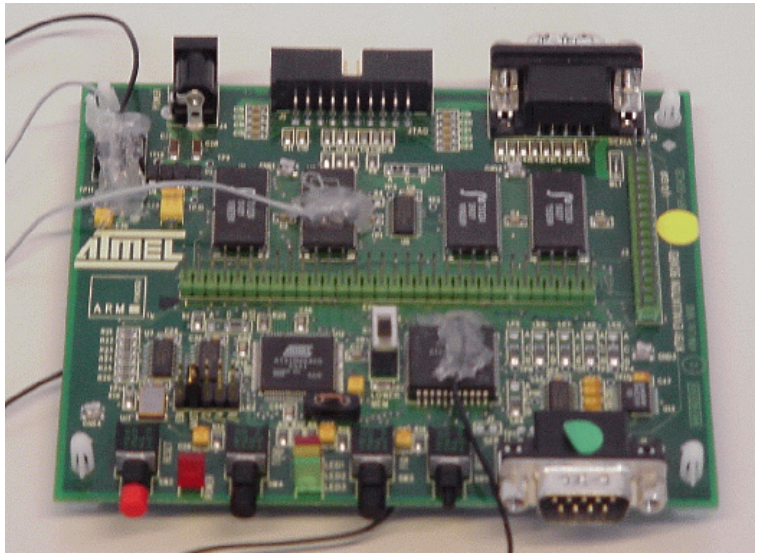


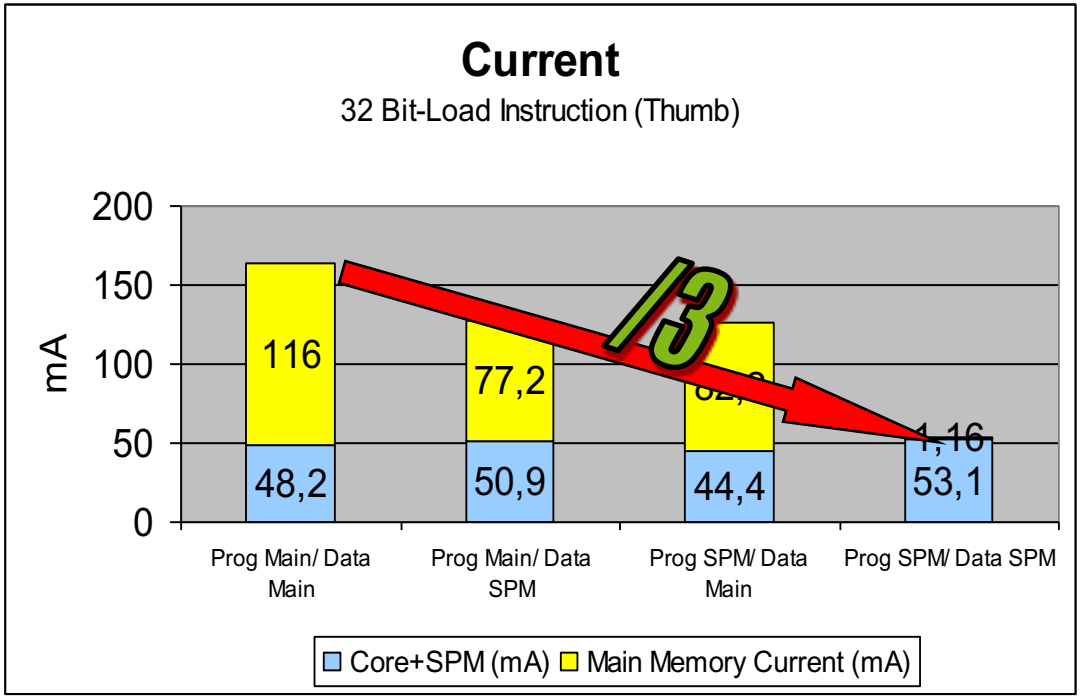
16 bit read; size in bytes;
65 nm for SRAM, 80 nm for DRAM

Source: Olivera Jovanovic, TU Dortmund, 2011

Vergleich mit Messungen

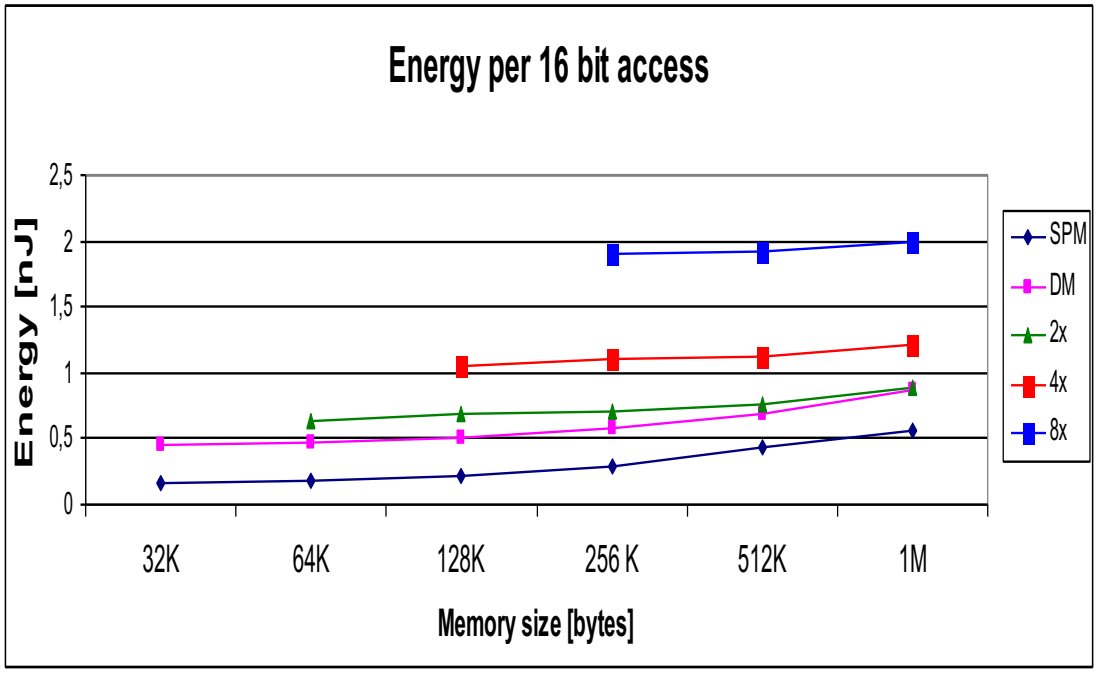
z.B.: ATMEL mit ARM7TDMI und ext. SRAM





Warum nicht einfach Cache benutzen?

- Großer Energieverbrauch von Tags, Komparatoren und Multiplexern



Cacti-basiert
- 16 Banken
- Hochperformanz
- 65 nm Technologie

O. Jovanovic, TU Dortmund, 2012

- Cache Kohärenz ist immer schwerer zu realisieren mit steigender Kernzahl

Zeitgenauigkeit (Predictability) und SPM

*… **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.*

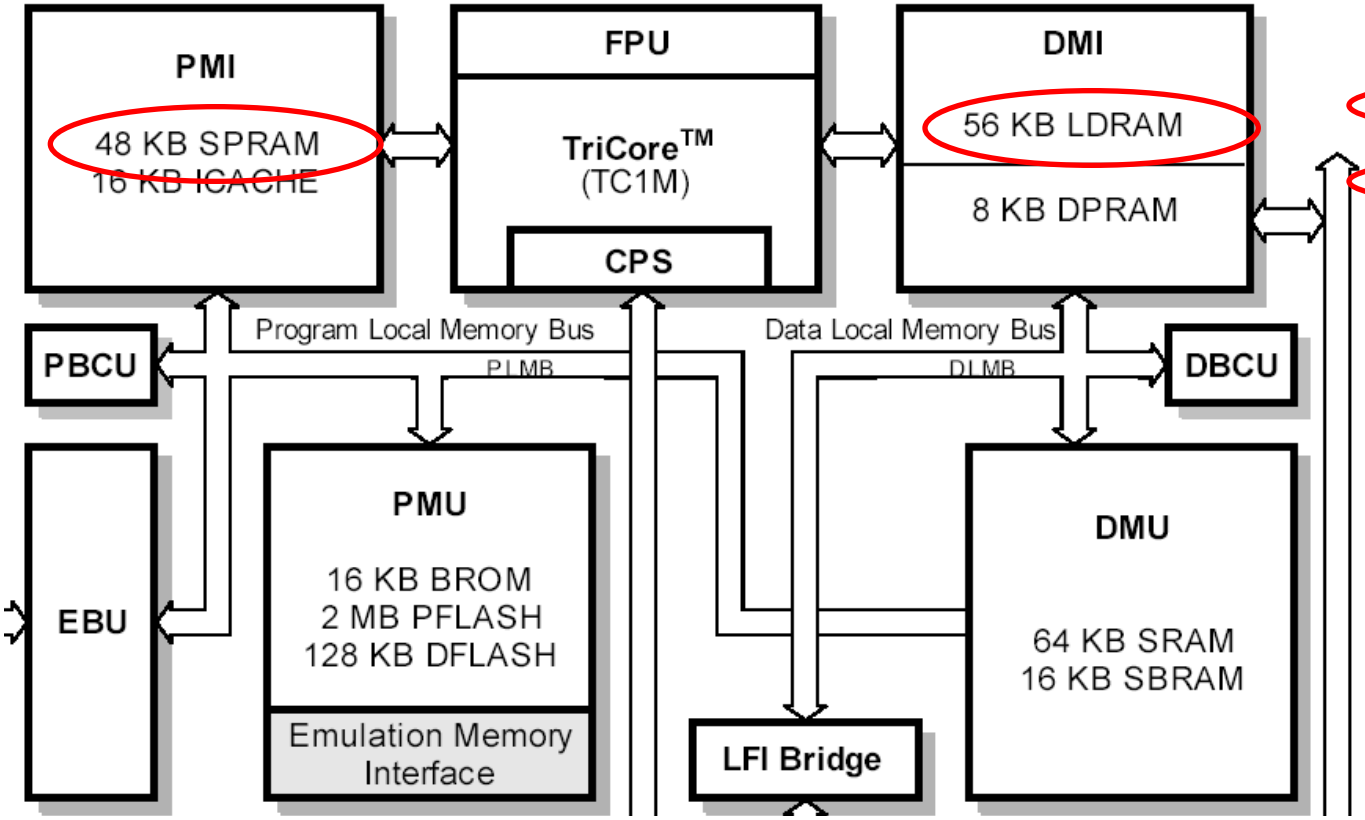
Edward Lee: Absolutely Positively on Time: What would it take?, IEEE Computer, 2005

Praxis von 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

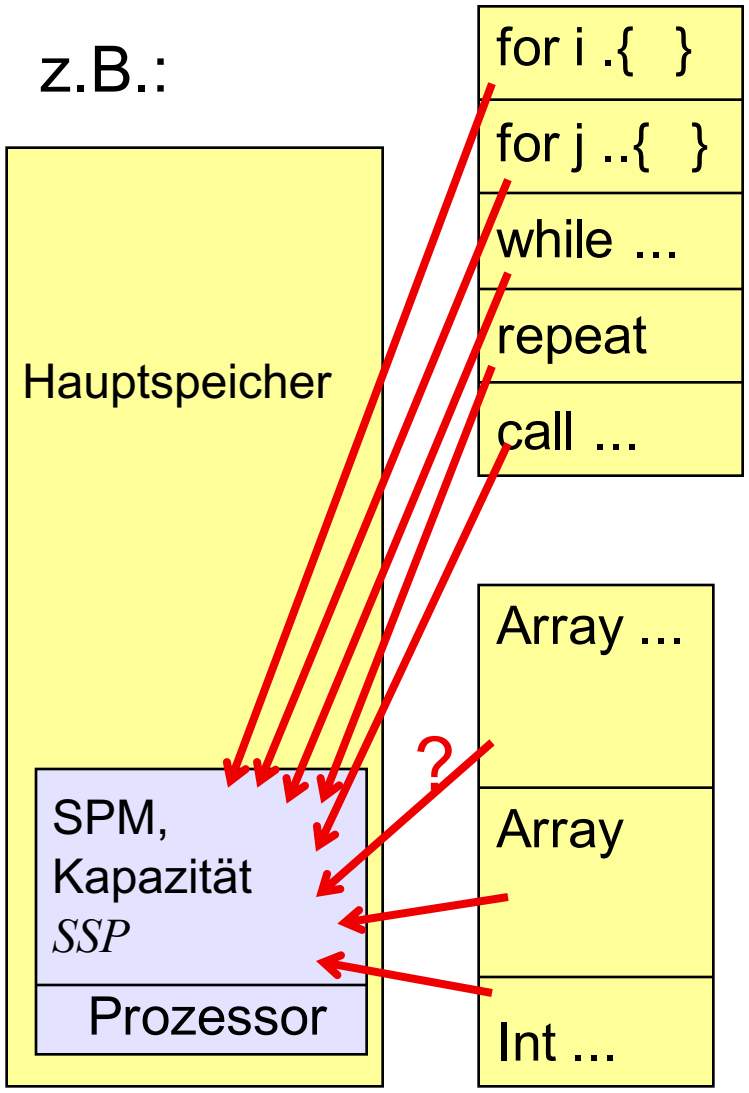


SPRAM:	Scratch-Pad RAM
ICACHE:	Instruction cache
LDRAM	Local data RAM
DPRAM:	Dual-port RAM
BROM:	Boot ROM
PFLASH:	Program Flash Mem
DFLASH:	Data Flash Memory
SBRAM:	Stand-by Data Mem
SRAM:	Data Memory
PRAM:	PCP Parameter Me
CMEM:	PCP Code Memory
PLMB:	Program Local Mem
DLMB:	Data Local Memory
RPB:	Remote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

Weitere 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

Optimierung von Daten und Befehlen in SPM



Welche Daten (array, loop, u.s.w.) sollen in SPM gespeichert werden?

statische Zuordnung:

Profit g_k und Größe s_k für jedes Objekt k. Maximieren Profit $G = \Sigma g_k$, innerhalb der Kapazität von SPM $SSP \geq \Sigma\ s_k$.

Lösung: Rucksackalgorithmus

dynamische Zuordnung: (nicht eingeschlossen) Die Daten und Befehle werden hin und her in SPM bewegt

Voraussetzung: **Integer linear programming (ILP)**

Zutaten:

- Kostenfunktion
- Beschränkungen

} mit linearem Ausdruck durch *integer* Variablen von einer Menge X

Kostenfunktion $C = \sum_i a_i x_i$ falls $a_i \in \mathbb{R}$, $x_i \in \mathbb{N}$ (1)

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

Def.: Die Kostenfunktion (1) zu minimisieren innerhalb der Beschränkungen (2) wird ein **integer linear programming (ILP) Problem** genannt.

Falls alle x_i entweder 0 oder 1 sind, wird das ILP Problem **0/1 integer linear programming Problem** genannt.

Voraussetzung: **Beispiel**

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

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

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

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 Ausdrücke –Kostenfunktion und Variablen –

Symbole:

$S(var_k)$ = Größe der Programmvariable var_k

$n(var_k)$ = Anzahl der Zugriffe auf Programmvariable var_k

$e(var_k)$ = **gesparte** Energie **pro Zugriff** auf Programmvariable var_k, wenn var_k im SPM Adressraum zugeordnet wird

$E(var_k)$ = **insgesamte gesparte** Energie, wenn var_k im SPM Adressraum zugeordnet wird (= $e(var_k)\, n(var_k)$)

$x(var_k)$ = Binärvariable, =1 falls var_k im SPM Adressraum zugeordnet wird,
=0 sonst

K = die Menge der Programmvariablen

ILP:

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

Innerhalb der Beschränkung

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

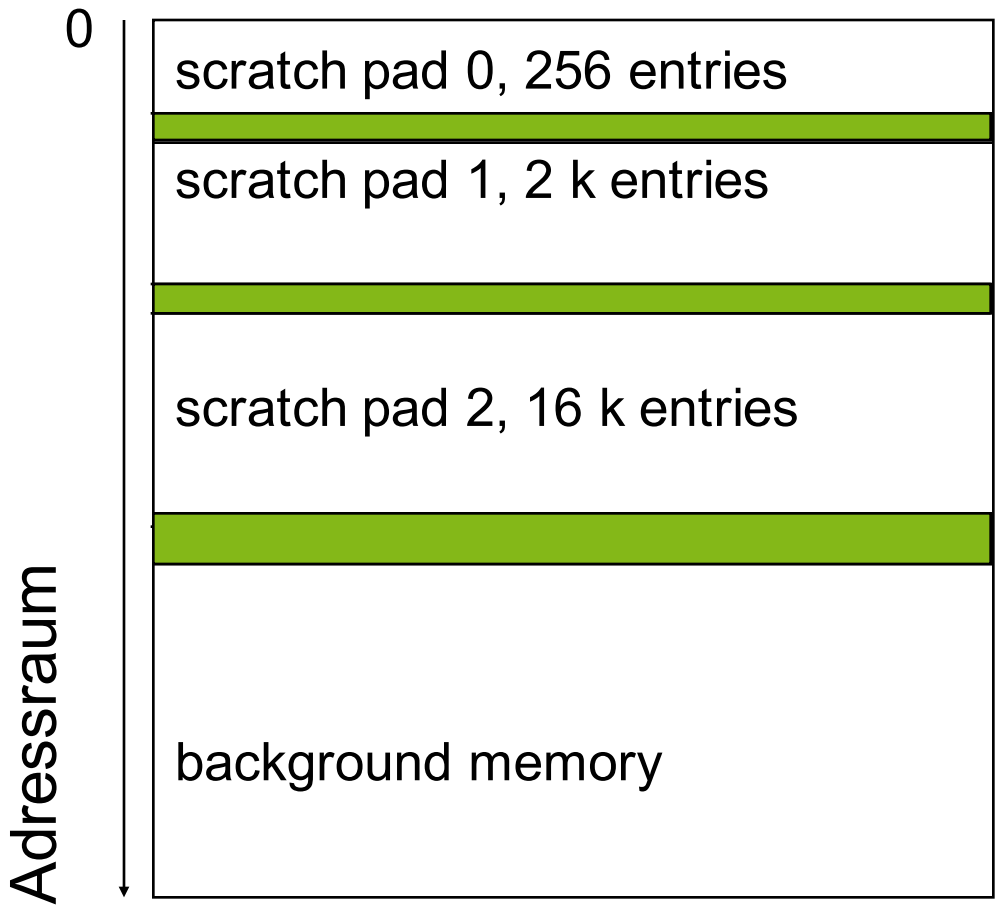
Ähnliche Definitionen für Programmfunktion F_i, $x(F_i)$, $E(F_i)$, und $S(F_i)$

Partitioning

Small is beautiful:

One small SPM is beautiful (☺).

Maybe several smaller SPMs are even more beautiful?



Beispiel: mögliche SPM Partitionen

Partitionen	Kombinationen						
	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

Beispiel für gesamte SPM Kapazität von 4096 Bytes

Optimierung mit mehreren SPMs

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

e_j: Energieverbrauch pro Zugriff auf Speicher j,
$x_{j,i}$= 1 wenn Objekt (Programmfunktion oder Programmvariable) i auf Speicher j zugeordnet wird, sonst ist es 0
n_i: Anzahl der Zugriffe auf Objekt i,
innerhalb der Beschränkungen:

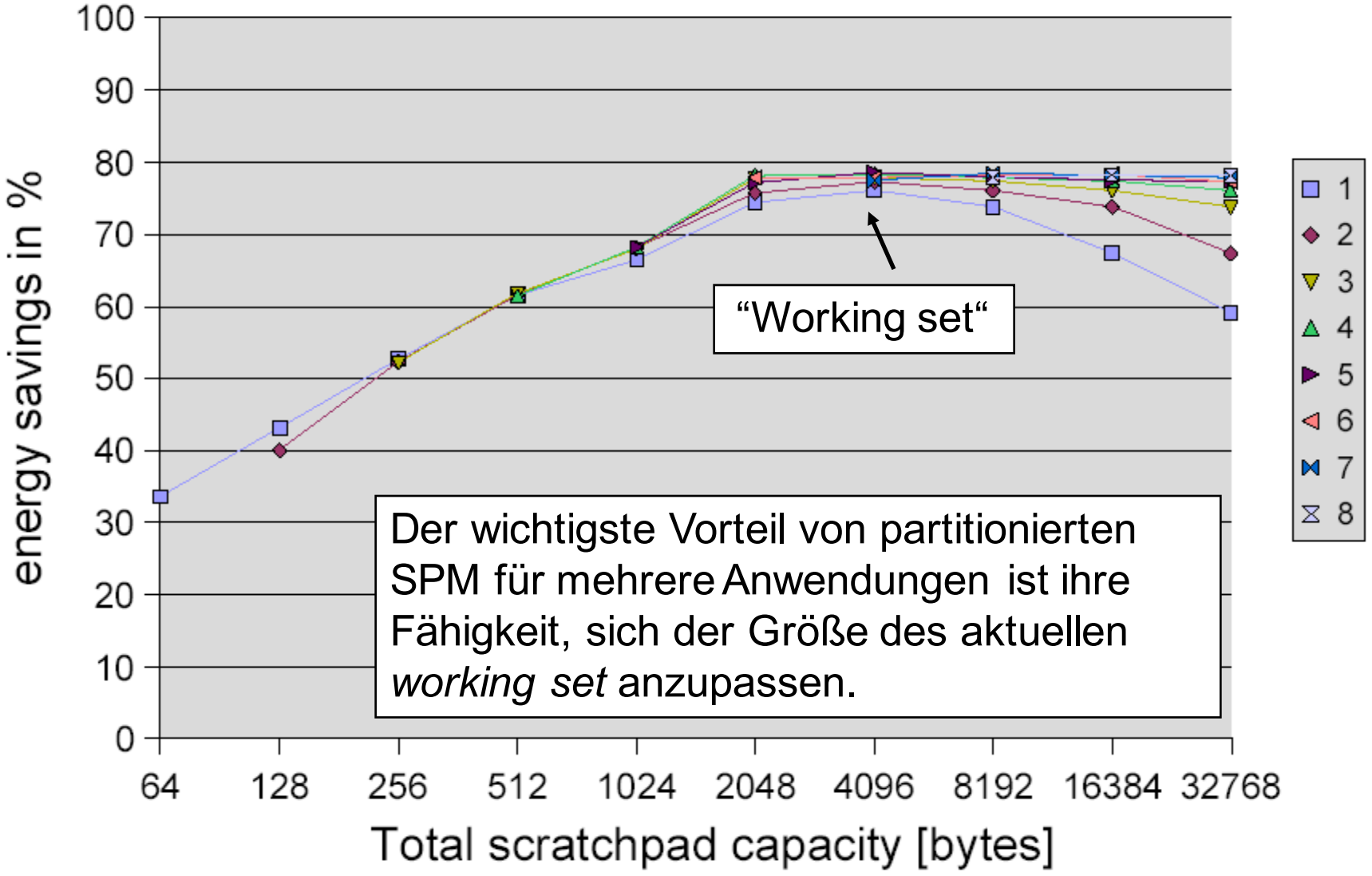
$$\forall j : \sum_i x_{j,i} \cdot S_i \leq SSP_j$$

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

Hauptspeicher wird als ein besonderer Fall von j betrachtet

S_i: Größe des Objekts i,
SSP_j: Größe von Speicher j.

Ergebnisse für (Anteil von) GSM coder/decoder



Wie viel Energieeffizienz kann man erreichen?

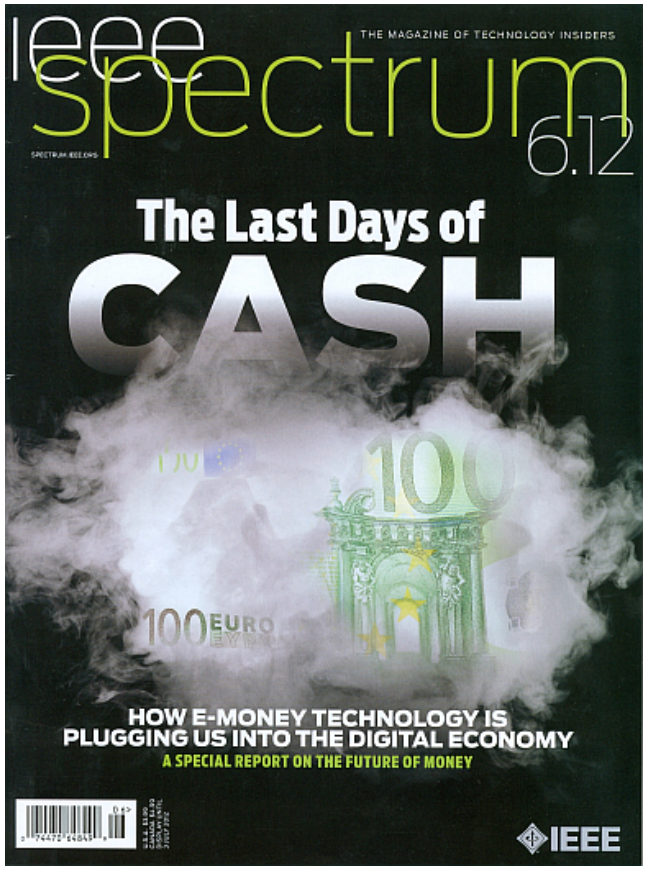
$$\text{new (normalized) run} - \text{time/energy consumption} = (1 - P) + \frac{P}{S}$$

- P: Anteil der Objekte, die man einem schnelleren oder energieeffizienteren Speicher zuordnen kann
- S: Verbesserung der Geschwindigkeit oder der Energieeffizienz

Es ist wichtig, nicht zu viele "nicht bewegbare" Objekte (1-P) zu haben, sonst bleibt die Energieeffizienz oder Performanz gleich sogar wenn $S \rightarrow \infty$

$$\text{improvement} = \frac{1}{(1 - P) + \frac{P}{S}} \quad \text{(Amdahl's law)}$$

Less seriously …



☞ Some people got already completely rid of cache