Rechnerarchitektur (RA)

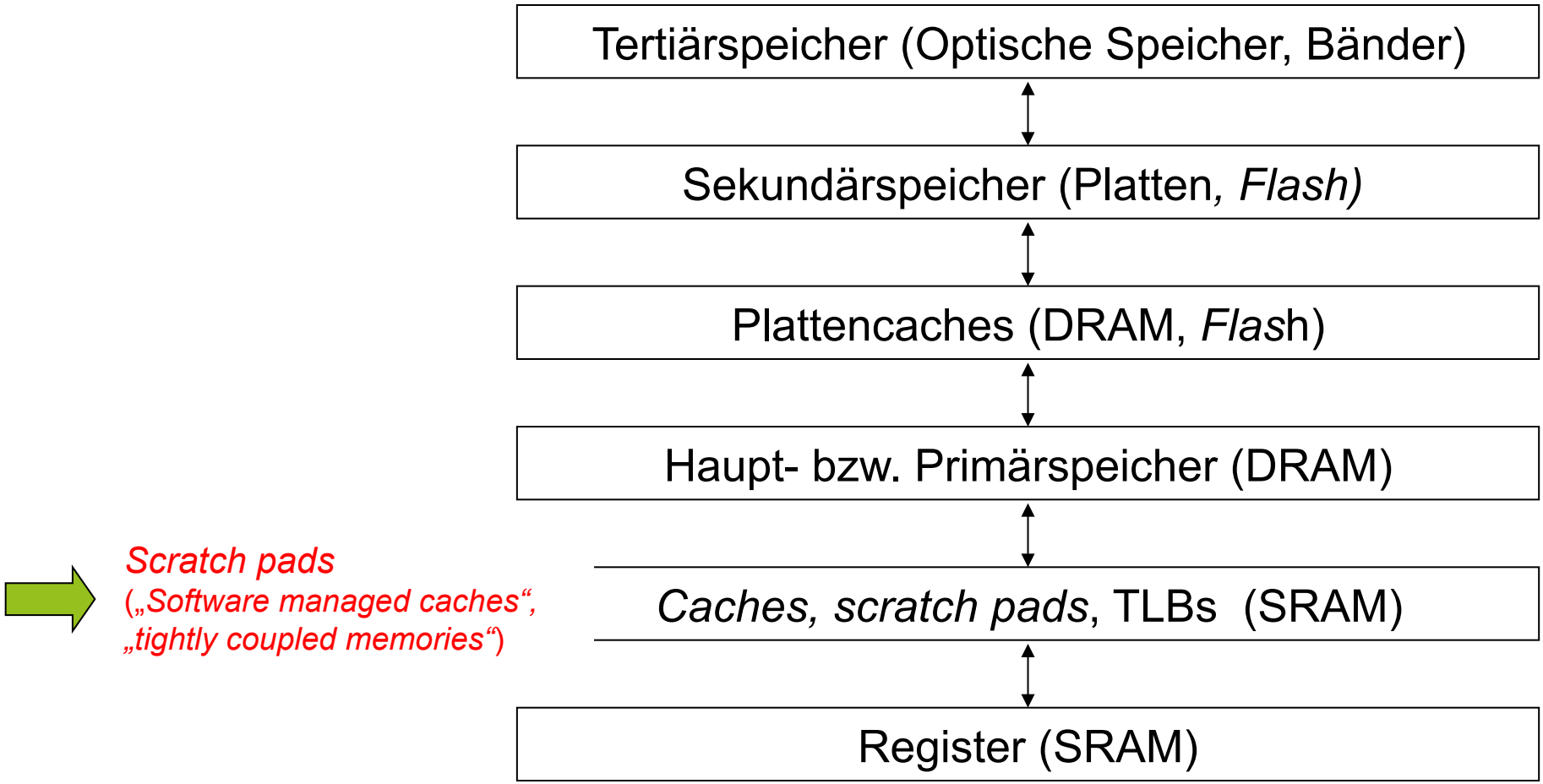
Sommersemester 2016

Scratchpad Speicher

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

2016/06/22

Mögliche Stufen der Speicherhierarchie
und derzeit eingesetzte Technologien



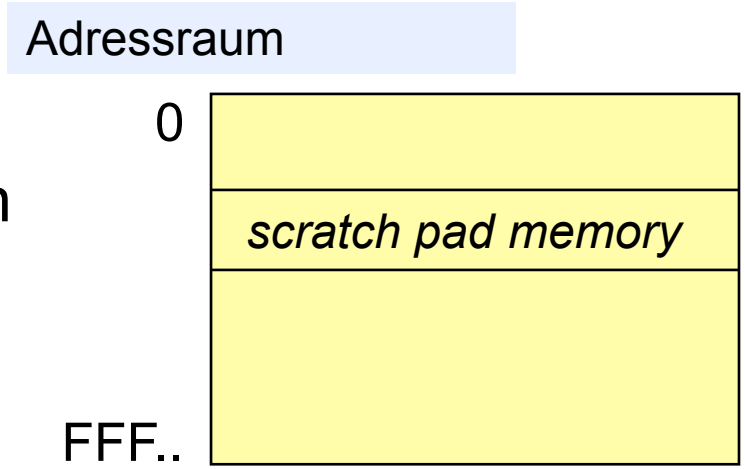
Scratch pad

Scratch pad?

© G. Marwedel, 2014

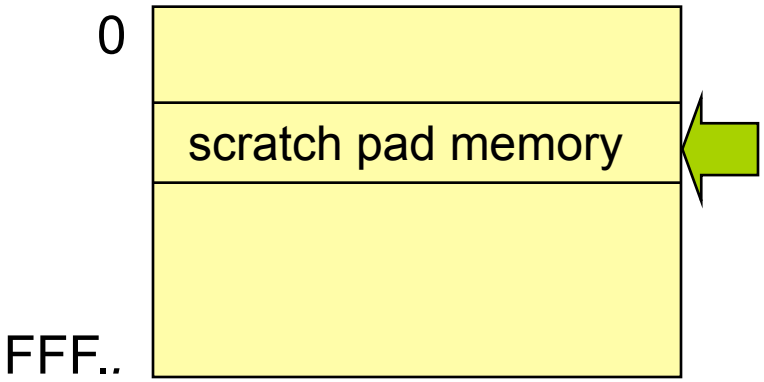
Scratch pad?

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



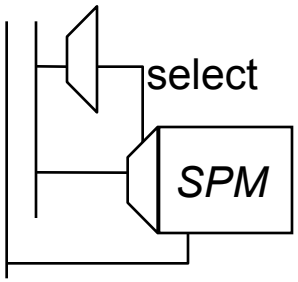
Scratch pad memories (SPM): Fast, energy-efficient, timing-predictable

Address space



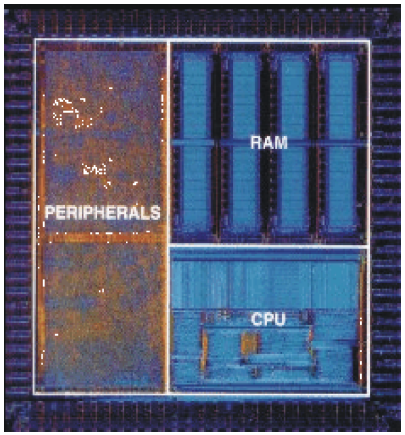
Small; no tag memory

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



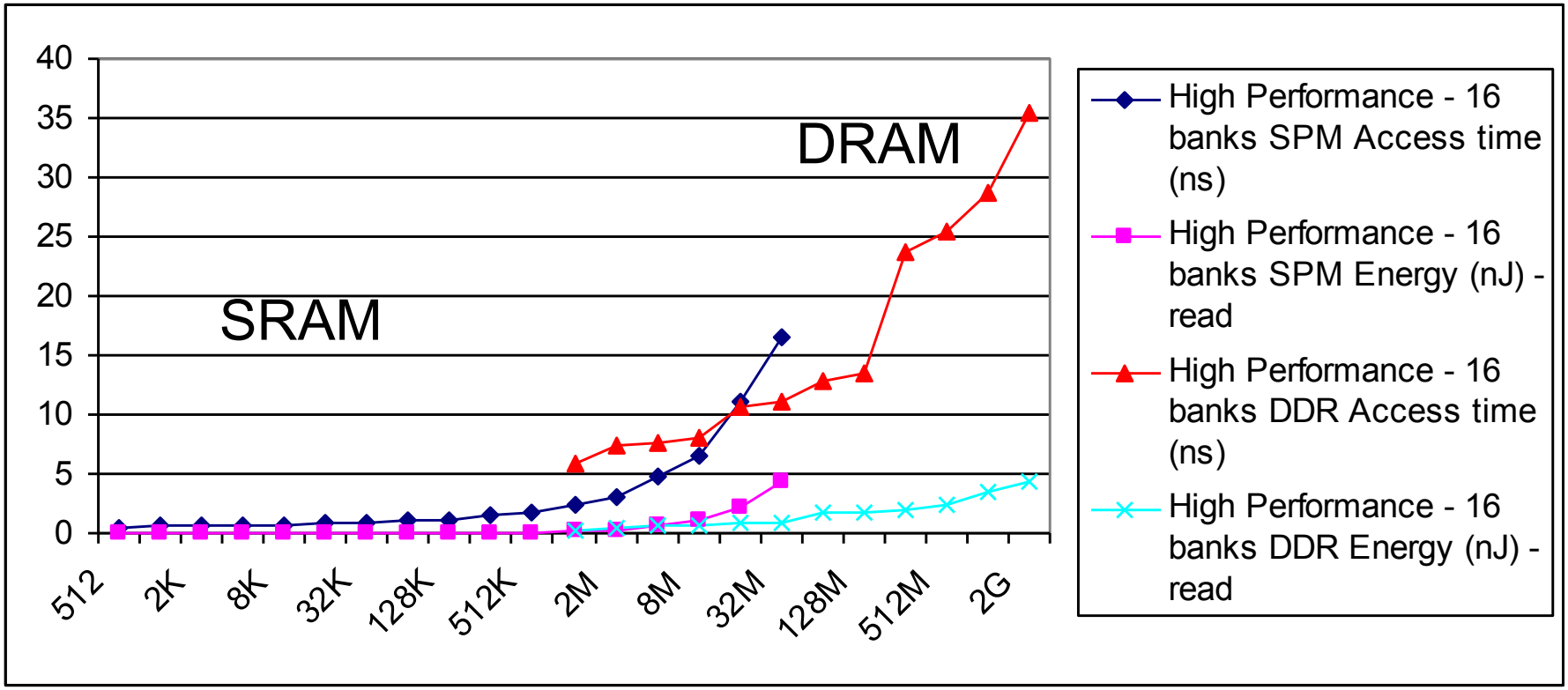
Example

ARM7TDMI cores, well-known for low power consumption



Energy consumption of memories

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

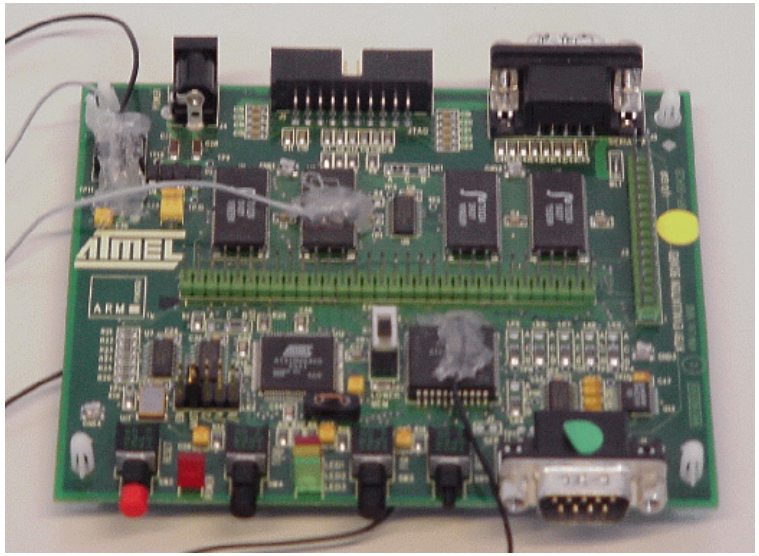


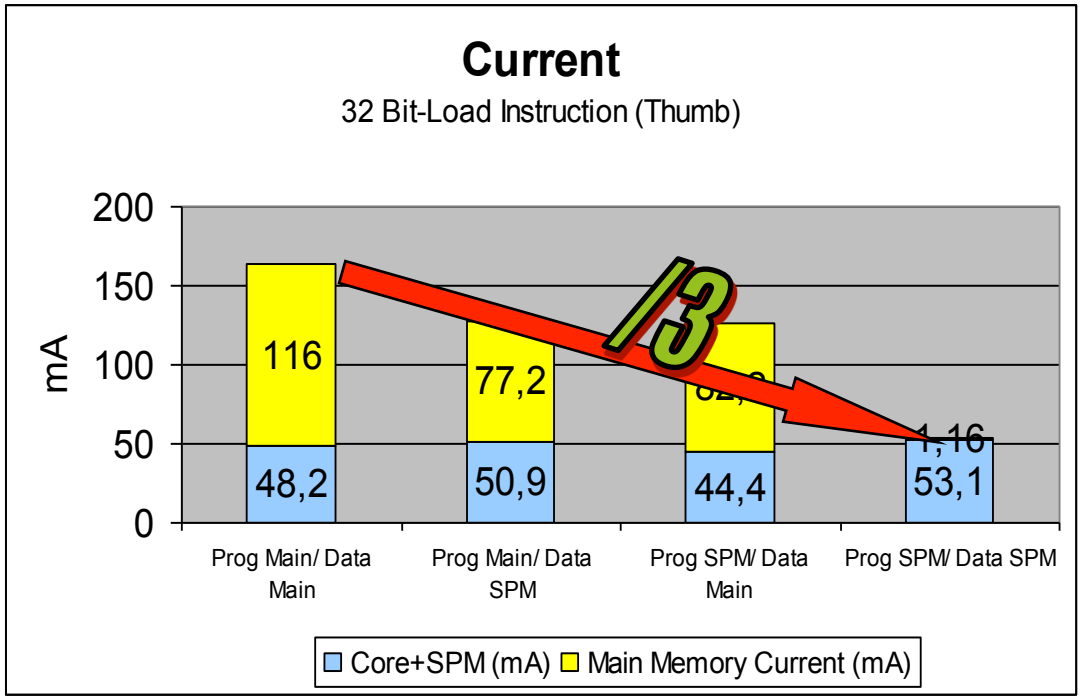
16 bit read; size in bytes;
65 nm for SRAM, 80 nm for DRAM

Source: Olivera Jovanovic, TU Dortmund, 2011

Comparison of currents using measurements

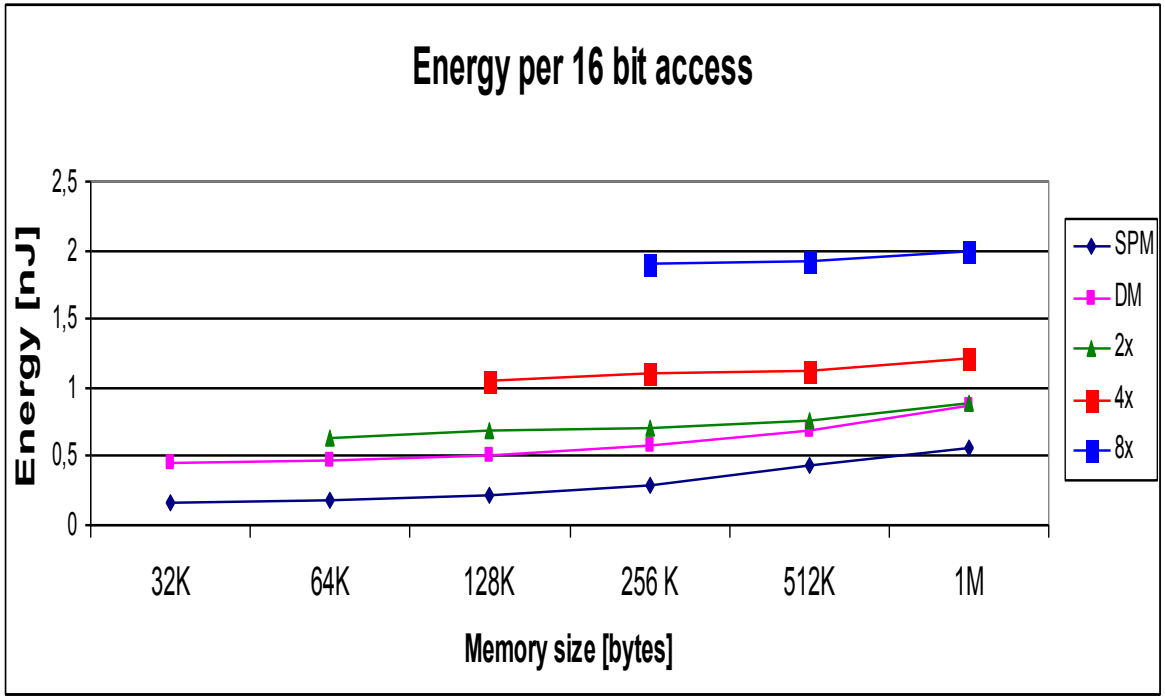
E.g.: ATMEL board with ARM7TDMI and ext. SRAM





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 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.*

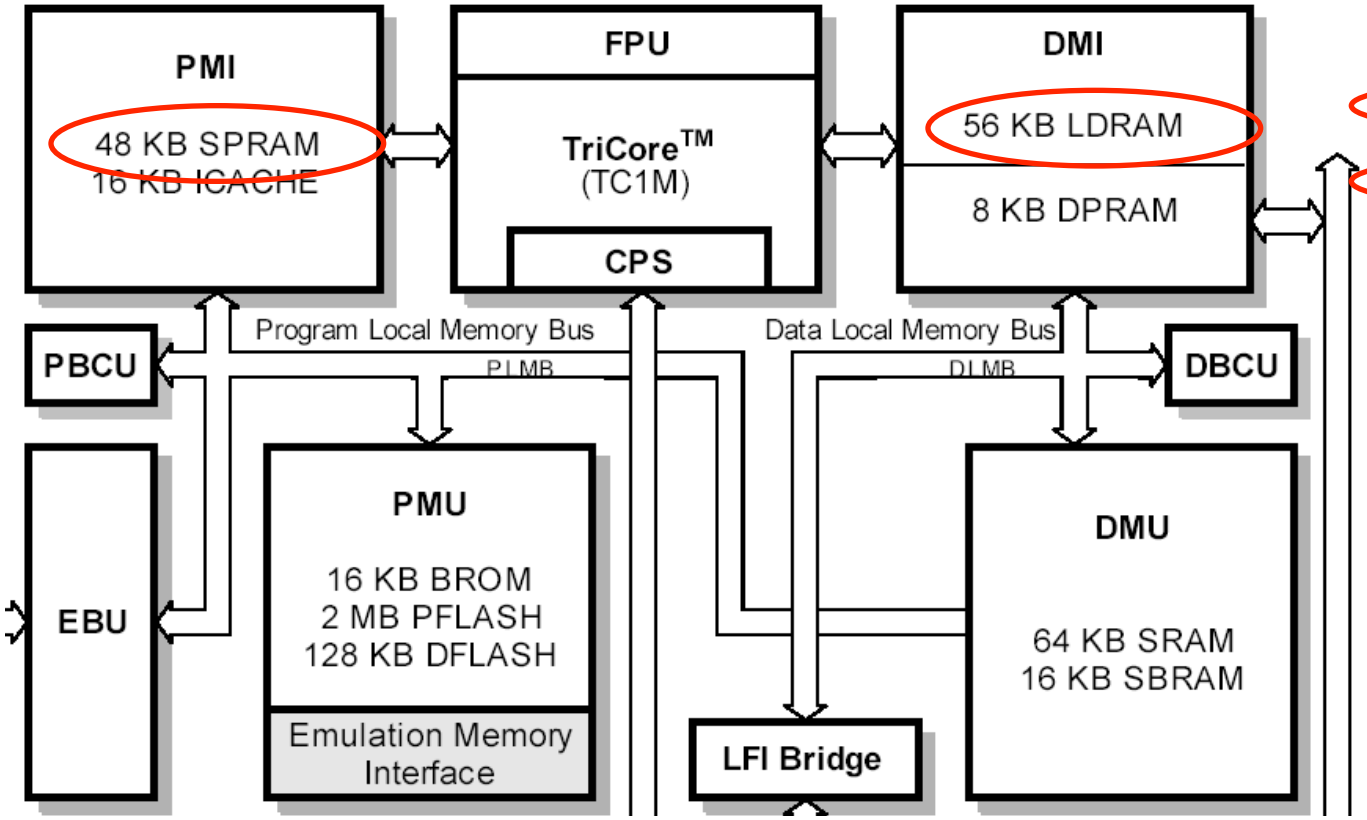
Edward Lee: Absolutely Positively on Time: What would it take?, IEEE Computer, 2005

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

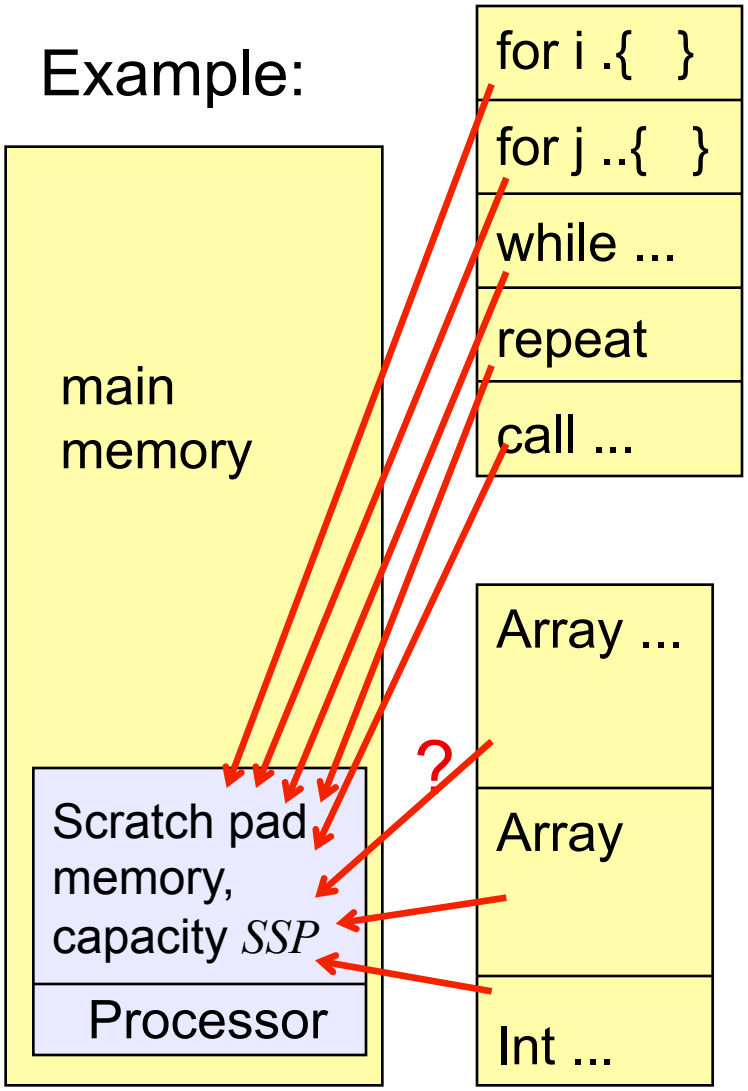
© Infineon, 2005

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)

Example:



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
1	1	0	11
1	1	1	15

← Optimal

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

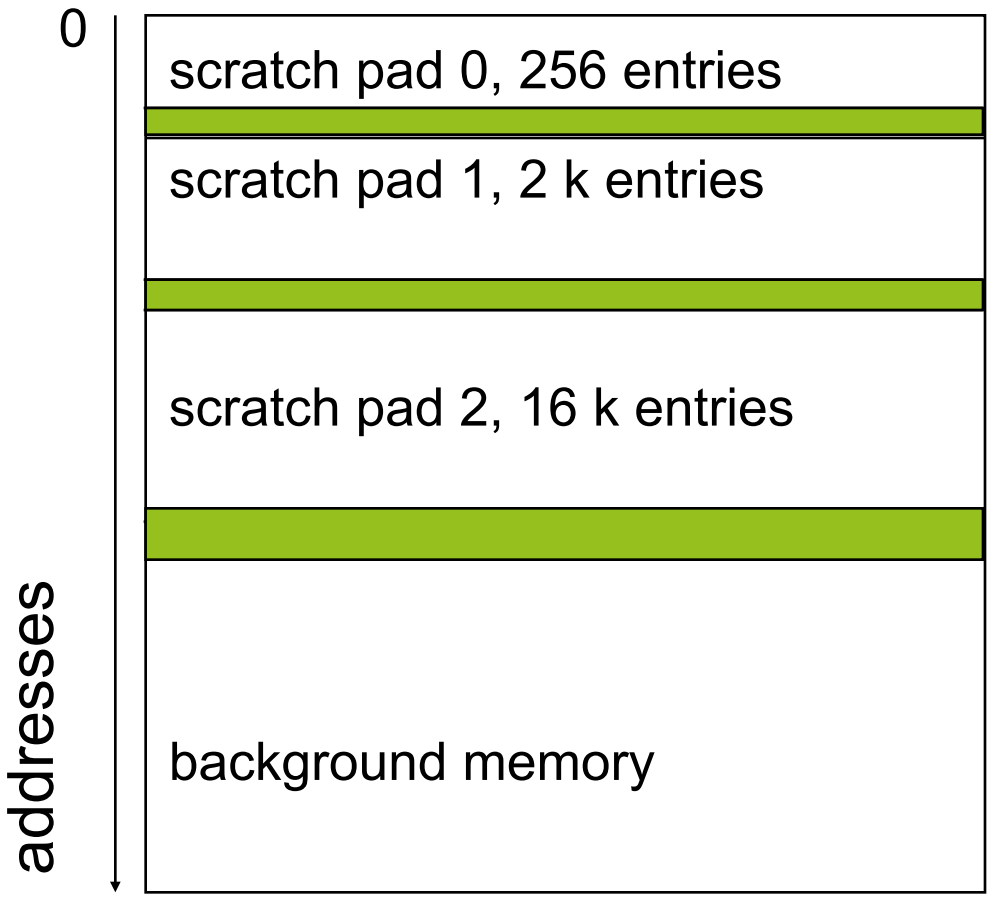
$\sum_{k \in K} S(var_k)\ x(var_k) + \sum_{i \in I} S(F_i)\ x(F_i) \leq SSP$

Partitioning

Small is beautiful:

One small SPM is beautiful (☺).

Maybe 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:

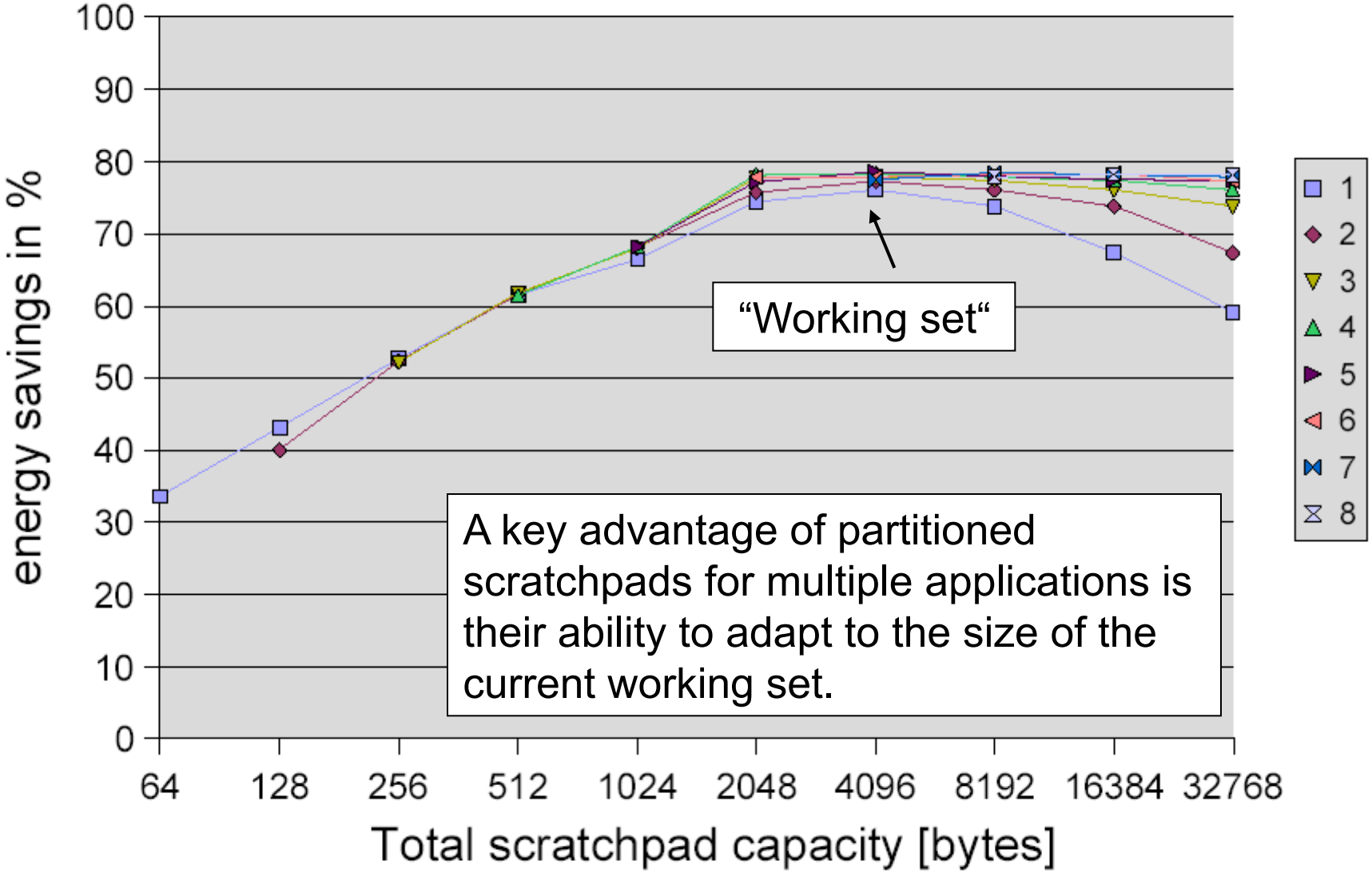
$$\forall j : \sum_i x_{j,i} \cdot S_i \leq SSP_j$$

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

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?

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

where

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

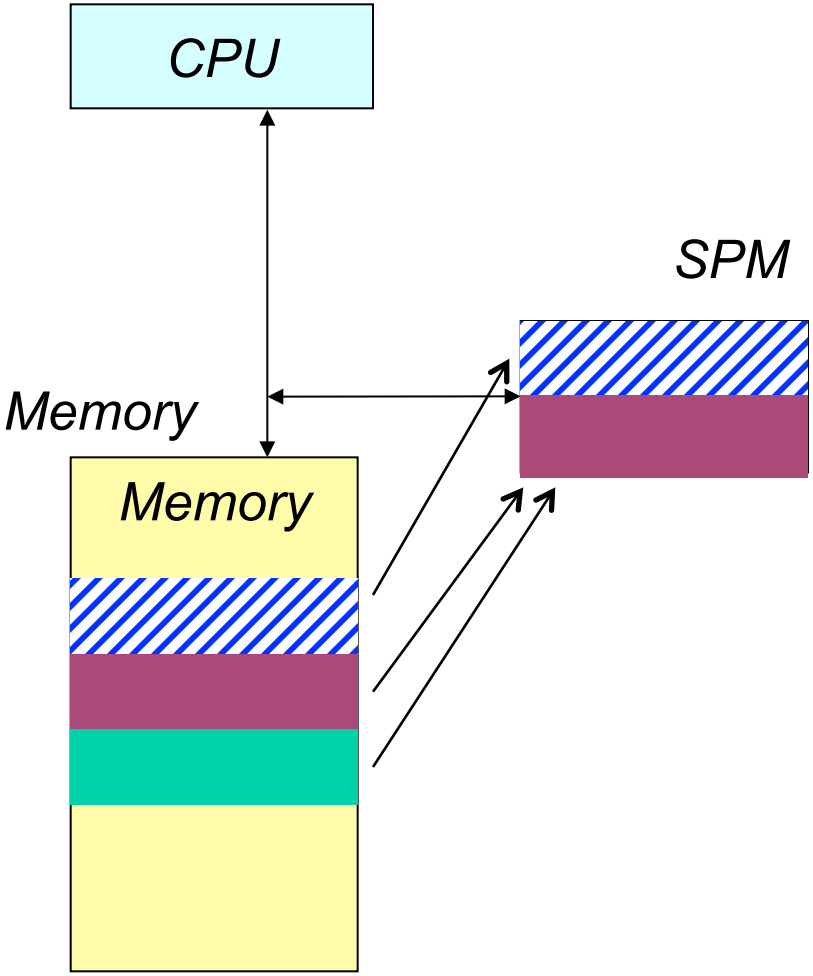
and

- S: speed/energy improvement

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

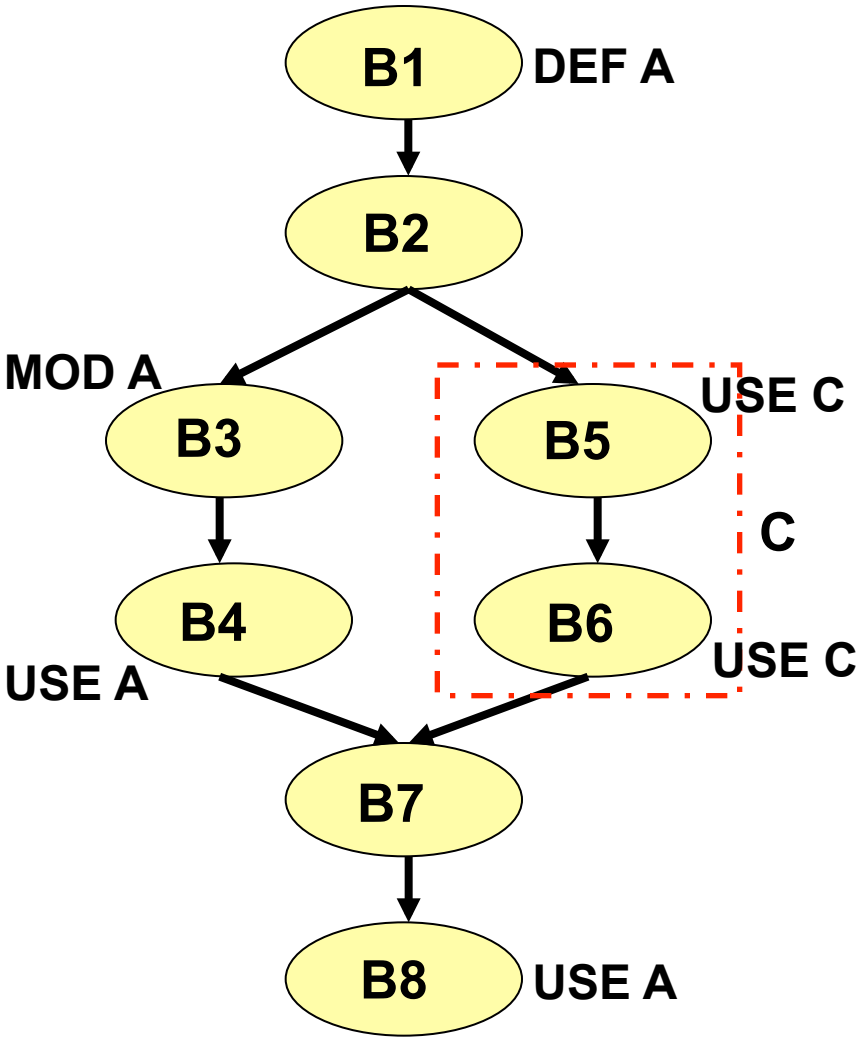
$$\text{improvement} = \frac{1}{(1-P) + \frac{P}{S}} \quad \text{(Amdahl's law)}$$

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

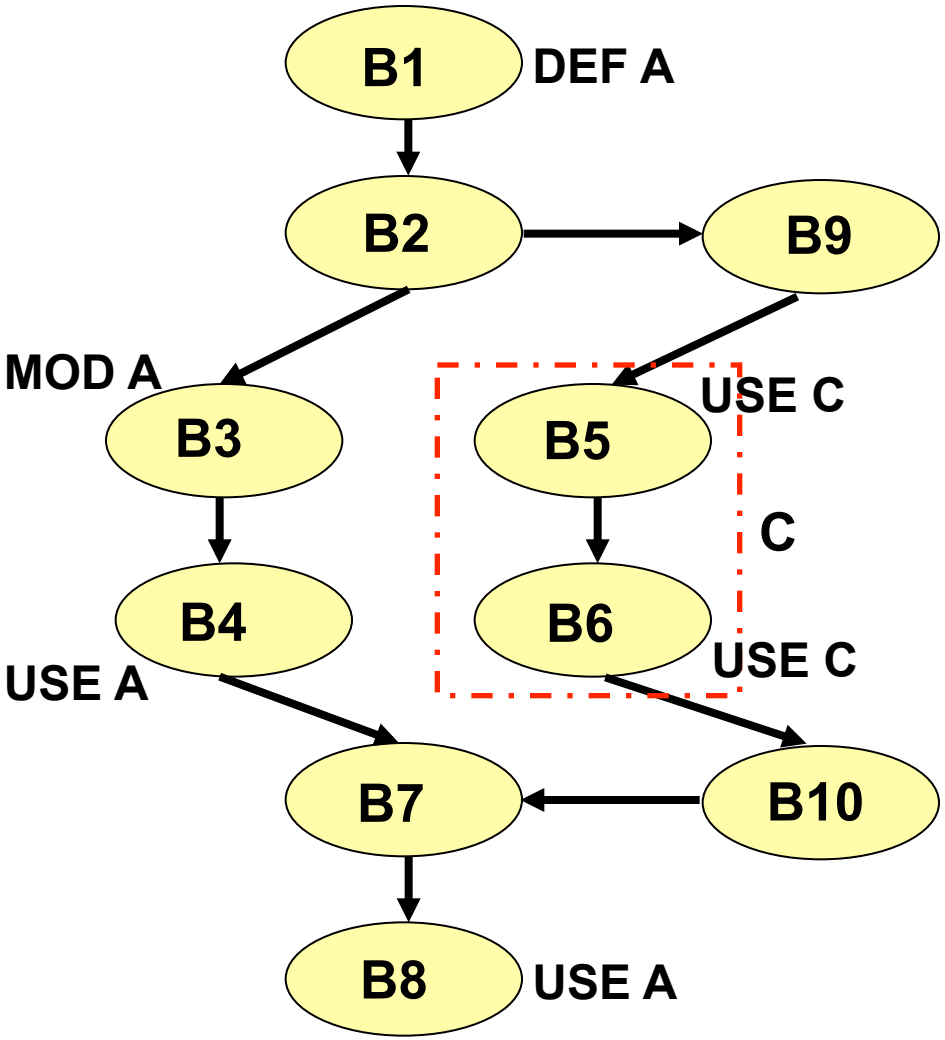
Overlaying allocation by Verma et al. (1)



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)



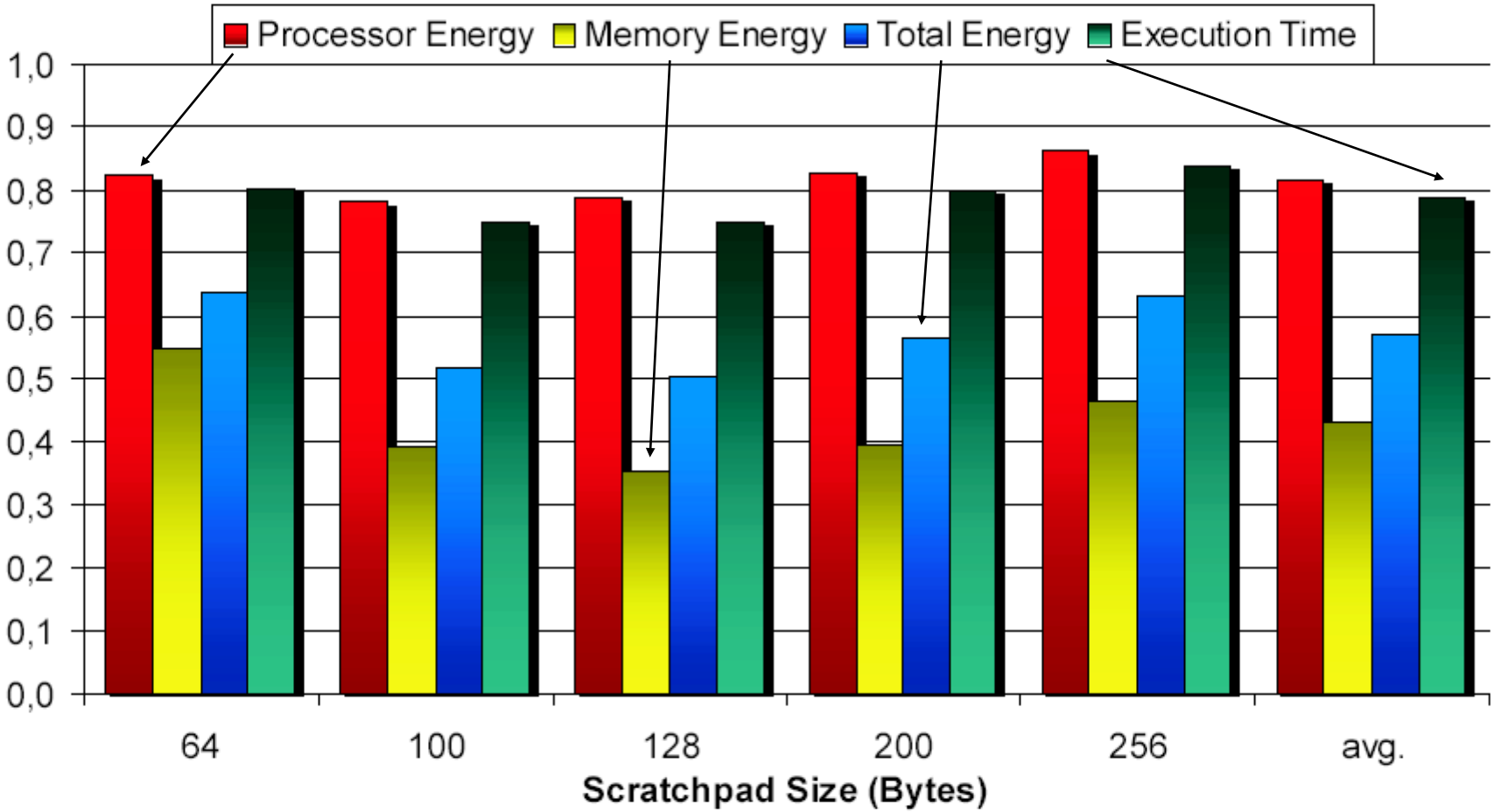

```
SPILL_STORE(A);
SPILL_LOAD(C);
```

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

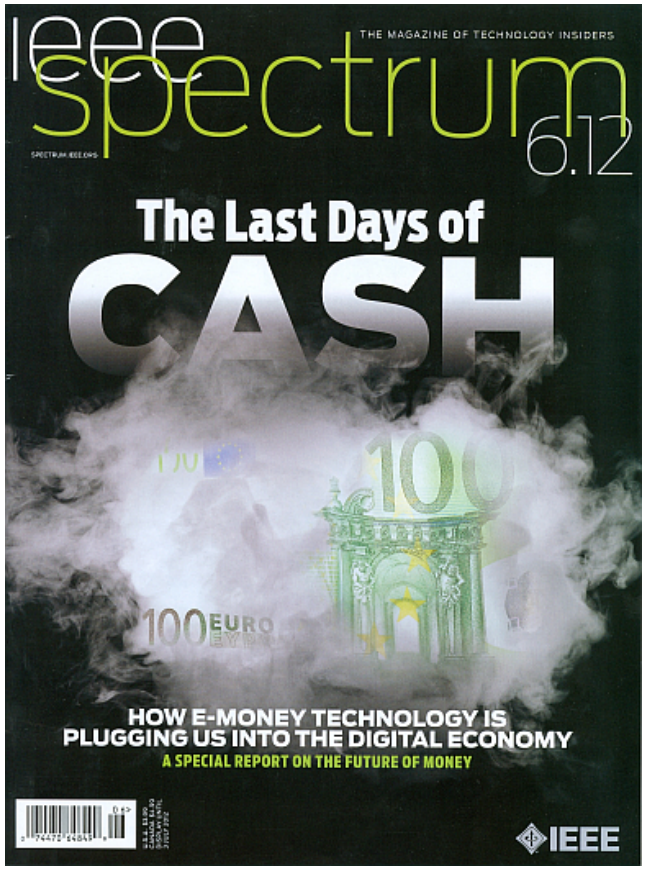
```
SPILL_LOAD(A);
```

Code handled like data

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



Less seriously …



© IEEE, 2012

☞ Some people got already completely rid of cache